

MATHEMATICAL MODELING IN INDUSTRY XI

ASSOCIATING EARTH-ORBITING OBJECTS DETECTED BY
ASTRONOMICAL TELESCOPES

Haseena Ahmed, Iowa State University
Prince Chidyagwai, University of Pittsburgh
Kun Gou, Texas A & M University
Yun Liu, University of Minnesota Twin Cities
Timur Milgrom, Clemson University
Vincent Quenneville-Bélair, McGill University

Mentor

Dr. Gary B. Green, The Aerospace Corporation

Institute for Mathematics and its Applications
August 17, 2007

Abstract

We are dealing with a problem of identifying streaks detected by a telescope of an earth orbiting object. The problem is reformulated into a clustering problem. A theoretical study is performed to show that the hierarchical algorithm fits the problem better than the k-means algorithm. The theory is tested through a series of experiments using Matlab routines for hierarchical clustering. The experiments result in conclusions that there needs to be theory created for choosing the cut-off parameter for the algorithm. Finally a section method is introduced for a future development of computationally efficient algorithms for large cardinality of the problem. The work completed gives a direction into what parts of the hierarchical algorithm need to be improved and how the cardinality of the problem needs to be handled.

Contents

1	Introduction	2
2	Problem Exposition	2
2.1	Orbit Space	2
2.2	Image Space	3
2.3	Hough Transform	4
3	Clustering Data	6
3.1	Similarity Functions	6
3.1.1	Distance Functions	6
3.1.2	Correlation	7
3.1.3	Weighted Distance Function	7
3.2	Orbit Space and Image Space Compared	8
3.3	Algorithms	8
3.3.1	Agglomerative Hierarchical Clustering Algorithm	9
3.3.2	Principal Direction Divisive Partitioning (PDDP)	10
3.3.3	<i>k</i> -means Clustering Algorithm	10
3.4	Implementation	14
3.4.1	Handling Large Data Set	14
4	Results	14
4.1	Clustering	15
4.2	Sectioning Method	17
4.3	Varying the value of cutoff in hierarchical clustering	18
4.4	<i>k-means</i> Clustering	18
5	Conclusion – Future Work	19

1 Introduction

Astronomical telescopes detect the passage of an earth-orbiting object through their field of view as a streak in an image. Advances in technology make it possible for a new generation of few-degree telescopes with gigapixel sensors to detect objects previously unseen. Over a period of months, many objects will pass through the field of view, some appearing many times. There are estimates of 100,000 objects of size of one centimeter or larger in orbit that will be detected by high resolution telescopes. A large field of view telescope may see 600 streaks a night. Most of these objects are space debris and some are active satellites, but all pose a hazard to operational satellites. There is interest within the space community to discover and track all these objects.

If the telescope sensor is properly instrumented, it is possible to obtain time-tagged pairs of sufficient resolution angle that relate the space object position to the sensor. With enough angle pairs, it is possible to estimate the position and velocity (the state) of the object, along with estimates of the uncertainties of these parameters. The workshop problem is to develop techniques to associate all the streaks made by each object. Streaks created by an object must somehow be associated with one another and disassociated from those made by other objects. One solution approach treats the state data as vectors in \mathbb{R}^6 and uses statistical clustering techniques for the association. A variation on this approach addresses physical properties of the orbits, sorting according to those least likely to change with small state energy variations.

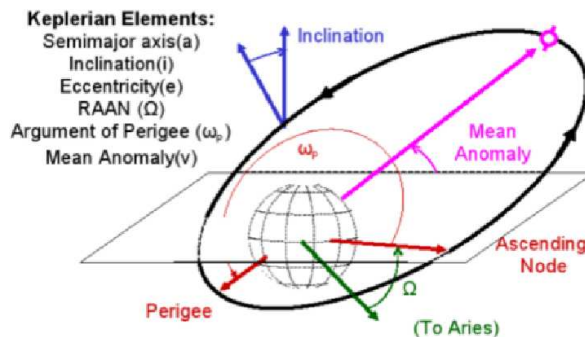
Regardless of the approach, there are several interesting aspects to the problem of dealing with streaks. Automatic streak detection is required, with transform techniques of interest. Orbit mechanics are essential to effective state estimation as well as clustering techniques. In addition, traditional clustering techniques are computationally taxing. A related problem is identification of asteroids that might pose a hazard to planet earth.

2 Problem Exposition

Once the observations from a telescope have been collected, our goal is to determine which streaks are due to objects which are already known. We consider two formulations of the problem; in each approach we try to solve the problem by comparing either the images generated by the streaks, or the orbits in which they move around the earth.

2.1 Orbit Space

In this approach we model each of the streaks by estimating the orbit of the satellites, and then comparing the parameters describing the orbit.



There are six keplerian orbital elements in this representation space. a is the semimajor axis of the elliptic orbit, referring to the size of the orbit. e is the eccentricity of the elliptic orbit, referring to the shape of the orbit. i is the inclination of the orbit plane with respect to the equatorial plane of the earth, which can be measured from the normal vector of the equatorial plane to the normal vector of the orbit plane. Ω represents the right ascension of the ascending node, or simply the **node**, referring to the angle in the equatorial plane measured positively from the axis to the location of the **ascending node**, which is the point on the equatorial plane at which the satellite crosses the equator from south to north. The argument of perigee, ω_p , is the angle measured from the ascending node to the perigee of the orbit. The last classical orbital element is the mean anomaly: it is the angle determining the satellite's current position relative to the location of perigee.

Now each vector of these six elements completely describes an orbit and is therefore well-defined. Actually the last element only provides the position of the satellite on the orbit, the first five elements define the orbit shape, size, and orientation. Since these elements play different roles in the motion of the satellites, we may assign different weights for them to measure their influence in the motion of the satellite.

2.2 Image Space

In this approach we model each of the streaks using information about its location and visual characteristics. In order to pinpoint the location of the streak we locate each point on the streak at a particular point in time by its *right ascension* and *declination*. These are two parameters that give us points in the coordinate system that is used by astronomers to locate objects in space. It is important to note that the length of the streak may be used to determine the importance of a particular vector in the data set; for example one might argue that streaks of negligible length should not have been considered in the cluster analysis of the vectors. Short streaks do not provide us with enough information to estimate the orbit accurately and thus propagating the orbits forward to the point in time when a new observation has been seen may result in large errors in the projected streak.

From the points that make up each streak, we are able to compute the angle made by the streak in right ascension and declination coordinate system and the length of the streak. We can identify each streak location by using, the midpoint of the streak. As a result, in this approach each streak is modeled by a vector in \mathbb{R}^4 .

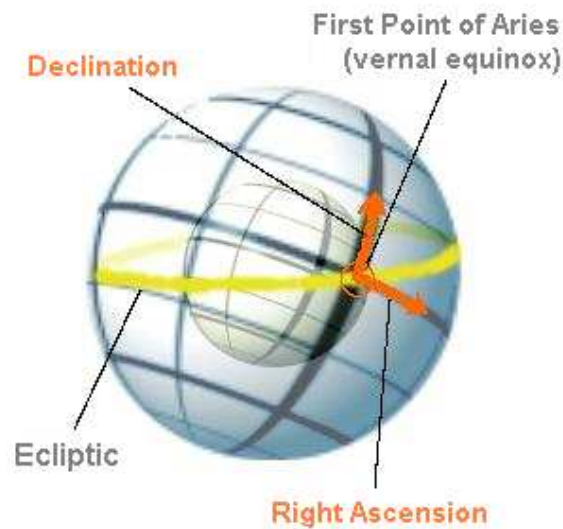
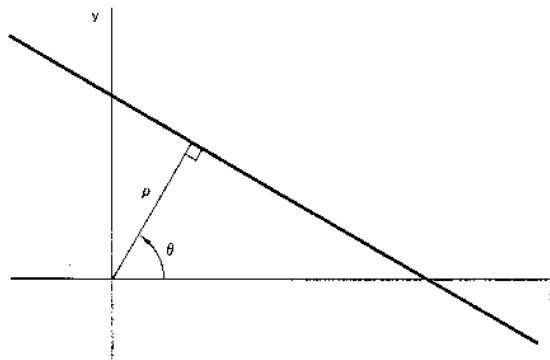


Figure 1: Right Ascension and Declination [12]

2.3 Hough Transform

Since the nature of our problem involves comparing simple images of straight edges, one of the approaches considered using the Hough transform. The Hough transform is a popular tool in digital image processing that allows detection of a straight edge in an image. Traditionally when we look at an image that has a straight edge, we interpret the straight edge in terms of its point slope intercept ($y = mx + b$). The Hough transform considers a line in terms of its distance from the origin represented by ρ and the angle of its normal vector to the origin represented by θ . It is graphically seen in Figure 2.

Figure 2: Line Representation [1]



The relation between the two formulations of the line can be stated as

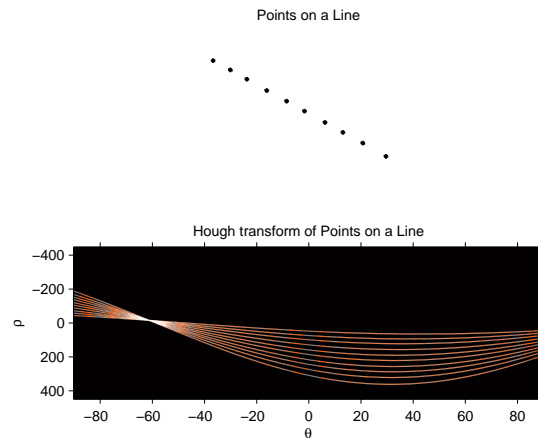
$$\rho = x \cos(\theta) + y \sin(\theta).$$

Based on this relation, the Hough transform transforms each point (x_0, y_0) of the line into a curve in the (ρ, θ) domain defined as a function by

$$\rho(\theta) = x_0 \cos(\theta) + y_0 \sin(\theta).$$

The graphical interpretation of how the transform works is provided in Figure 3. In Figure

Figure 3: Line Representation



3 we have a plot of points that are all on the same line. After taking the transform of those points, we get a set of curves in the (ρ, θ) domain. It is easy to notice that all the curves in the Hough domain go through the same point that identifies the value of ρ and θ that correspond to the line. If we would take more points, all located in-between the points we originally used, we will generate curves in between the top and the bottom curve of the original set of curves. Similarly if we choose points that are outside the original set of points, more curves will be generated outside those already drawn. From this experiment we see that the Hough transform can be used to identify a line segment in a digital image by giving the length and the location of the line segment in terms of the (ρ, θ) variables.

The Hough transform is useful as far as recognition of lines in image processing. The problem that we are dealing with involves knowing everything about the line requires comparison of lines. We reached the conclusion that the comparison of the lines would make more sense using the standard line representation rather than the Hough transform of the line because the Hough transform does not provide any new information of the line while it takes away the easy visualization of the line.

3 Clustering Data

Data clustering [10, 3, 5] is a method of creating groups of objects, or clusters, in such a way that objects in one cluster are very similar and are distinct from objects in different clusters. In our problem, we want to cluster streaks made by the same object or satellite. In the literature of data clustering [10, 3, 5], similarity measures, similarity coefficients, dissimilarity measures, or distances are used to describe quantitatively the similarity or dissimilarity of two clusters.

The method of clustering depends mainly on the data set available. Data can be binary, discrete, or continuous. See [3] for classification of data-types. In our case, the data comes from streaks detected by astronomical telescopes. Typically 600 streaks are observed for an average 10 hour each night, and the amount of time for which the readings are taken varies depending on the application we are interested in. If data is collected for a year, then we have about 200,000 streaks that need to be clustered.

3.1 Similarity Functions

3.1.1 Distance Functions

In order to apply a clustering algorithm to the data, a similarity function needs to be defined on some space of parameters. In this case, the parameters can lie in the orbit space or in the image space.

The intuition is maybe stronger in the image space since it relies on easily visualized streaks and seems to be based on an Euclidean distance. Hence, the first distance that could be implemented might as well be the Euclidean distance or some similar L^p norm. In order to scale each parameter, the parameters are normalized according to their average and their standard deviation. It is then possible to apply some type of weighting; as a first guess, an arbitrary one can be tried. Another type of similarity function that could be implemented on the image space is the area between the two streaks divided by the average of their length.

For the orbit space, a standard L^p norm will be attempted on the normalized set of vectors. However, some parameters might also have different importance and so some weighting should be considered. This weighting could be attempted with a comparison of the energy required to change each parameter. Indeed, the inclination of the orbiting plane of a satellite, for instance, is much harder to vary than its mean anomaly. Another similarity function that could be set up includes the area or the volume in between the two orbits in analogy with the image space case. Finally, the orbit space could be transformed so that it refers to Cartesian coordinates by using, instead of angles, the foci and a third point on the ellipse that is not collinear in order to define the ellipse. The reason to do so is to avoid the discontinuous behavior of the keplerian elements.

3.1.2 Correlation

Many researchers have noted the importance of standardizing variables for multivariate analysis. Otherwise, variables measured at different scales do not contribute equally to the analysis. The appropriate standardization method depends on the data set and the conventions of the particular field of study. One example of a paper that discusses standardization is “Metric and Euclidean properties of dissimilarity coefficients” [4]. In addition, Milligan and Cooper presented an in-depth examination of standardization of variables when using Euclidean distance as the dissimilarity metric [8]. In this problem, we use standardization technique as follows

$$V' = (V - \min V) / (\max V - \min V),$$

where V represents the value of the variable in the original data set. This method allows variables to have differing means and standard deviations but equal ranges. In this case, there is at least one observed value at the 0 and 1 endpoint. For example: for the sequence 28, 2, 76, 54, 45, 8, we have the maximal number as 76 and the minimal one as 2. After standardization, we get a new sequence .351, 0, 1, .703, .581, .081 respectively.

In this problem, there are totally six Keplerian parameters describing the orbit that created a streak. They are independent, so the weights in them are just the same. We put weight 1 to every parameter. We omit this simple weight expression.

Then we use Euclidean distance for any two streaks:

$$d(i, j) = \sqrt{\frac{(\alpha_1 - \alpha_2)^2 + (e_1 - e_2)^2 + (i_1 - i_2)^2}{+(\Omega_1 - \Omega_2)^2 + (\omega_{p_1} - \omega_{p_2})^2 + (M_1 - M_2)^2}}$$

all the variables in this formula are standardized. Then we can get the dissimilarity matrix as follows:

$$M = (d_{ij})_{n \times n}$$

where n is the number of streaks.

This matrix is symmetric and the elements on the diagonal line are all zero. This matrix is very helpful, since it is key for the later steps.

3.1.3 Weighted Distance Function

In the implementation of the clustering, an important factor is to choose the proper similarity function. The general L^2 norm doesn't tell the difference of these elements of the vectors that how they affect the total energy of the satellite. Now we take a weighted Euclidean distance which can do better in it.

From the Trajectory equation we can tell that the term $\xi = \frac{v^2}{2} - \frac{\mu}{r}$ is constant, where $\mu = G(M + m)$, G is a universal constant, M is the mass of earth, m is the mass of satellite. Hence the total energy of the satellite $E = \frac{1}{2}mv^2 - m\frac{\mu}{r} = m\xi$ is also constant. Using the identities $p = a(1 - e^2) = \frac{h^2}{\mu}$, where a is the semimajor axis of the elliptic orbit, e is the eccentricity of the orbit, and $h = r \times v$, we can get $\xi = -\frac{\mu}{2a}$. It's obvious that the energy is inverse proportional to a . Note $a = c/e$, it's easy to see the energy is proportional to e ,

and it's not depends on the other three elements at all. If we want to change the energy by $1/10$, then e need to be changed by $1/10$, also. And by an easy computation we can see that a need to be changed by $1/11$. This provides a way to assign proper weights to each elements: we can assign a weight of $1/11$ to a , $1/10$ to e , etc.

We can see from the numerical results that the results get much more accurate by weighted Euclidean distance.

3.2 Orbit Space and Image Space Compared

The streaks in our database were made in the past. In order to compare these streaks with a new streak at a new time, we first predict where the satellites that made the earlier streaks will be at the time of the new streak. If the same satellite made an older streak, the streak it makes at the time of the new streak should match the new streak. A similar hypothesis can be made for orbital parameters – the orbital parameters arising from a new streak should match those predicted at the new streak time from old orbit parameters. After modeling the streaks in the image space or orbit space, our problem reduces to cluster the resulting vectors. We normalize all the vectors by scaling the mean and standard deviation of each parameter. We then compute a standardized figure for each of the elements. This is an approximation because we obtain the average and standard deviation from the streaks that end up in the same field of view after forward propagation in time. This sample might not be a good representation of the population of streaks as it might not be large enough. Normalizing is done to minimize the bias in weighting which may result from the different measuring scales and ranges. We did not have enough time to study equations relating the energy needed to change each of the parameters in order to obtain an alternative weighting scheme. We then apply different clustering methods to the normalized vectors.

Cluster analysis requires a similarity or dissimilarity function, in the case of our model we use the distance between the vectors of a given pair. In the case of unperturbed data containing several streaks from unique object we use an L^2 -norm in the clustering algorithm. We use a weighted Euclidean norm in the case of perturbed data.

3.3 Algorithms

In general, conventional clustering algorithms can be classified into two categories: hierarchical algorithms and partitional algorithms. Hierarchical algorithms have a tree structure where two closest clusters are merged at each step, whereas partitioning algorithms create a one-level non-overlapping partitioning, where elements are assigned to clusters at one stage depending on the distance between elements in a data set. Our problem has two factors in deciding the type of clustering is that it would be able to group a large data set, and secondly, it should be able to identify clusters that might have only one element. There are a number of other issues associated with clustering, e.g., finding clusters in data where there are clusters of different density or where the data has noise or outliers.

We study three popular methods for grouping data; the agglomerative and divisive hierarchical clustering method and the k -means clustering method. We will compare these

methods by showing their advantages and disadvantages.

3.3.1 Agglomerative Hierarchical Clustering Algorithm

Hierarchical clustering uses two basic approaches. In agglomerative clustering, we start with the points as individual clusters and, at each step, merge the closest pair of clusters. The merge or split is made based on a dissimilarity or similarity function. The algorithm can be given from [10] as follows:

Input: Data set D with m elements, dimension n , convergence criterion.
Grouping Phase: Compute the proximity matrix, if necessary
repeat
 merge the closest two clusters;
 update the proximity matrix to reflect the proximity between
 the new cluster and the original clusters;
until, stopping criteria is reached.

The proximity matrix gives a measure of dissimilarity between clusters. Variants of the agglomerative clustering are obtained by the way in which the dissimilarity between clusters is defined. The most common ones [10, 3] are the single link method, where the closest neighbor distance is used, complete link, using the farthest neighbor distance, and group average method, using the average distance between all possible pairs of data points that are made up of one data point from each group.

The hierarchical clustering algorithms have some disadvantages. The data points that have been incorrectly grouped at an early stage cannot be reallocated. Also, different similarity measures for measuring the similarity between clusters can lead to different results, so it is important to choose a good similarity measure.

For large data sets such as ours, hierarchical methods become impractical unless other techniques are incorporated, because usually hierarchical methods are $O(n^2)$ for memory space and $O(n^2 \log n)$ for CPU time [10], where n is the number of data points in a data set.

Agglomerative hierarchical clustering algorithms make good local decisions about combining two clusters since they can use information about the pairwise similarity of all points. However, once a decision is made to merge two clusters, it cannot be undone at a later time. This approach may prevent obtaining a global optimum.

There can be difficulties using hierarchical clustering when the relative sizes of the clusters to be merged are large. One is the weighted approach which treats all clusters equally and another is the unweighted approach, wherein the number of points in each cluster is taken into account. In other words, treating clusters of unequal size equally gives different weights to the points in different clusters, while taking the cluster size into account gives points in different clusters with same weight.

3.3.2 Principal Direction Divisive Partitioning (PDDP)

PDDP [7] is a clustering algorithm developed using techniques from numerical linear algebra. It is a method which recursively divides the data into smaller and smaller clusters, assembling all the clusters into a binary tree. Starting with the root node representing the entire dataset, PDDP computes the hyperplane which best divides the data. All the data on one side of the hyperplane is associated with one branch, and the data on the other side of the hyperplane is associated with the other branch. The process continues on each branch until some stopping criteria is met.

Algorithm PDDP.

1. **Start** with $n \times m$ matrix \mathbf{M} of vectors, one for each data sample, and a desired number of clusters k_f .
2. **Initialize** Binary Tree with a single Root Node
3. **For** $c=2,3,\dots,k_f$ **do**
4. **Select** leaf node C with largest *ScatterValue*, and L and R :=left and right children of C
5. **Compute** $\mathbf{v}_c = g_c(\mathbf{M}_c) \equiv u_c^T(\mathbf{M}_c - \mathbf{w}_c e^T)$
6. **For** $i \in C$, if $v_i \leq 0$, then assign data sample i to L, else assign it to R.
7. **Result:** A binary tree with k_f leaf nodes forming a partitioning of the entire data set.

Remark: PDDP is designed to be applied to the entire data set at once, and for good performance requirements the entire data set be present in core memory. A variant of PDDP has been developed to handle large data that cannot fit into memory at once. The piecemeal PDDP breaks the original data into sections which will fit into memory, and clusters these sections individually. The cluster centers are used to create approximations to the original data. The piecemeal PDDP is able to take advantage of the approximations.

3.3.3 k -means Clustering Algorithm

The k -means algorithm is classified as a partitional or non-hierarchical clustering method. In this algorithm, the number of clusters k is assumed to be fixed and known *a-priori*. There is an error function in this algorithm that measures how closely the vectors are clustered. It proceeds, for a given initial number of k clusters, by allocating the remaining data to the nearest clusters and then repeatedly changing the membership of the clusters according to the nearest clusters and then repeatedly changing the membership of the clusters according to the error function until the error function does not change significantly or the membership of the clusters no longer changes. The algorithm can be given as follows: Let

D be the data set with m elements, and let $C_1, C_2, C_3, \dots, C_k$ be the k disjoint clusters of D . The error function is defined as

$$Error = \sum_{i=1}^k \sum_{\mathbf{x} \in C_i} d(\mathbf{x}, \mu(C_i)), \quad (1)$$

where $\mu(C_i)$ is the centroid of the cluster C_i , $d(\mathbf{x}, \mu(C_i))$ is the distance between x and $\mu(C_i)$, and we use a distance measure for $d(\mathbf{x}, \mu(C_i))$. The cluster centroids are recalculated either after each instance assignment, or after the whole cycle of re-assignments. The two main phases of the algorithm are the initialization phase and the iteration phase. The algorithm can be given from [3] as follows:

Input: Data set D with m elements, k clusters, n dimensions.
Initialization Phase: Choose a set of k instances as centers of the clusters, C_i is the i th cluster.
Iteration Phase:
repeat
 d_{ij} = distance between element i and cluster j ;
 $n_i = \arg \min_{1 \leq j \leq k} d_{ij}$;
 Assign element i to cluster n_i ;
 Recompute the cluster means of any changed clusters above;
until, no further changes of cluster membership occur in a complete iteration.

Given two different sets of clusters that are produced by two different runs of K-means, we prefer the one with the smallest error given by equation (1) since this means that the prototypes (centroids) of this clustering are a better representation of the points in their cluster.

The space requirements for k -means are modest because only the data points and centroids are stored. Specifically, the storage required [3] is $O((m+k)d)$ where m is the number of points and d is the number of dimensions. The time requirements for k -means are also modest: essentially, linear in the number of data points. In particular, the time required is $O(I * k * m * d)$, where I is the number of iterations required for convergence. I is often small and can usually be safely bounded, as most changes typically occur in the first few iterations.

The k -means algorithm is very efficient for clustering large data sets and high-dimensional data sets. Here, the clusters are formed by using a center-based algorithm, which is designed to cluster numerical data in which each cluster has a center (spherical data).

The iterative approach is appealing because it allows the possibility for a poor initial clustering to be corrected in later stages. It has several drawbacks, however. In many cases, the results from a k -clustering are not unique and they depend on the initial selection of k centers, and the algorithm is known to converge to a local optimum and not a global

optimum. To minimize this problem, the algorithm is usually run several times and the best clustering is chosen from the different runs.

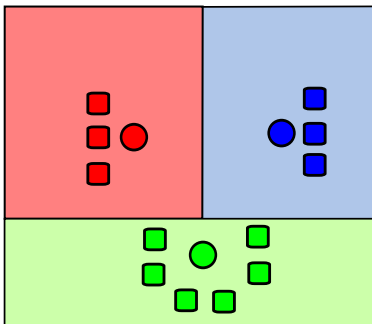


Figure 4: Three optimal clusters

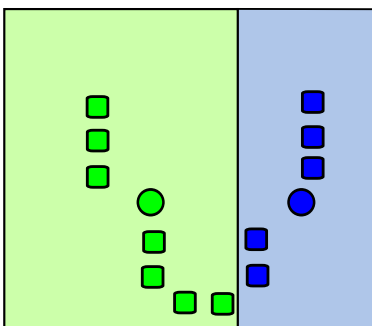


Figure 5: Two wrong clusters because of bad choice of k

Some of the features of k -means clustering, which can be advantageous or disadvantageous depending on the data to be clustered, are given below.

Finding optimal number of clusters The k -means clustering requires giving the number of clusters k *a-priori*. This can be a drawback for clustering problems for which the number of clusters is not known since choosing a wrong value of k may impose structure on the data set and hide the true structure. Figure 4 shows a picture of three optimal accurate clusters color coded in red, blue and green. The squares represent the data elements and circles represent the centers. Figure 5 shows what happens when we choose to cluster data into two sets.

There is no generally accepted procedure for determining the number of clusters. In 2000, Pelleg and Moore ([9] proposed a x -means algorithm wherein the Schwarz criterion is used globally and locally in order to find the best number of clusters k . Given a data set D , a family of alternative models $M_j = \{C_1, C_2, \dots, C_k\}$, the *a-posteriori* probabilities

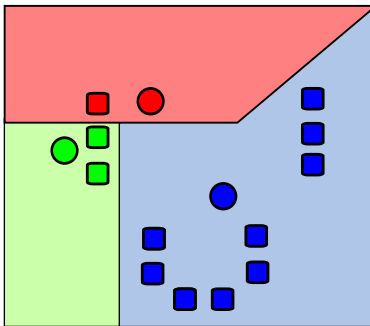


Figure 6: Three wrong clusters because of bad choice of initial centers

are used to score the models. The Schwarz criterion, BIC, is defined as

$$BIC(M_j) = l_j(D) - \frac{p_j}{2} \log n,$$

where $l_j(D)$ is the loglikelihood of D according to the j th model and taken at the maximum likelihood points, and p_j is the number of parameters in M_j . The model with the largest score is selected. The implementation of this idea is given in [9, 3].

Optimal initialization of centers of clusters The clustering results and convergence speed of the k -means algorithm are dependent on the initial centers and hence, their selection is an important issue in the k -means algorithm. Incorrect initialization of centers will lead to an incorrect clustering result since the centroids defined may not be located at the centers of the corresponding clusters. Instead, they are either at some boundary points among different clusters or at points biased from some cluster centers. This is illustrated in Figures 4 and 6. If the initial cluster centers are chosen as circles given in Figure 6, notice that the clusters formed are incorrect, and are given by the shaded region.

A commonly used approach for initializing cluster centers is the direct method [3], where, the idea is to choose the first k distinct objects as initial centers. Another approach involves choosing the initial centers as widely spread as possible. This is done by choosing the first center at random, then choosing the next center as the data element that is farthest from the first, then choosing the next center as the data element farthest from the first two centers and so on. In this way, we obtain a set of initial centroids that is guaranteed to be not only randomly selected but also well separated.

Clusters with one element/Presence of outliers Outliers can unduly influence the clusters that are found. Because of this, it is often important to discover outliers and eliminate them beforehand. However there are clustering applications such as ours for which outliers should not be eliminated, since one outlier (streak) could be produced by a satellite which orbits the earth infrequently.

There are variants of the k -means clustering approach. They differ in election of the initial k means, dissimilarity calculations, strategies to calculate cluster means.

3.4 Implementation

3.4.1 Handling Large Data Set

As mentioned previously, about 200,000 streaks need to be clustered to deal with the data collected in a year. Because of the size of the problem, the techniques of clustering large data sets must be applied here to discover natural groups of streaks and orbits and hence identify distinct satellites. [6] provides some techniques dealing with this.

For the two different clustering algorithms we studied, i.e., the agglomerative hierarchical clustering and the k-mean clustering methods, different techniques are developed to handling large data sets.

For the Hierarchical Agglomerative Clustering Methods(HACMs) [6] there are four main methods: single link, complete link, group average link, and Ward's method. These algorithm are fairly similar, differing mostly how they calculate the similarity between two clusters. The complexity is $O(n^2 \log n)$, which is impractical for large data sets. The following algorithms improve the viability of HACMs for large data sets.

Scatter/Gather: The data can be divided into a small number of clusters first, after a short summary, those clusters are gathered, clustered, and the process repeats.

Buckshot Algorithm: Take a random sample of the data size \sqrt{kn} , where k is the number of clusters and n is the total number of documents. Then find k "centers" in the sample using links such as single or group average link. Assign each to one of the clusters, e.g.. by closest distance. Note cluster centers may shift, so repeat the assignment until a quality metric is satisfied.

Fractionation Algorithm: Divide the data set into N/m groups of fixed size m , with $m > k$, then cluster the data in each of the groups using some clustering algorithm. Consider these N/m groups as individuals, and repeat the process, until only k groups remain. Assign each datum to one of the k clusters, as in buckshot. Then repeat assignment as needed.

Fractionation takes longer time than buckshot, although asymptotic complexity is the same. According to the results of the partitioning, fractionation seems to make better clusters.

4 Results

We developed a test bed to evaluate the various techniques presented in earlier sections of this report. A program written by Gary Green simulated the pointing history of an astronomical telescope scheduled for operation in 2013. This program accepts as input a catalog of satellites, propagates the satellites during a time period of interest, and determines which satellites pass through the field of view of the telescope. (Not all satellites are seen by the telescope.) It computes the right ascension/declination history of each

streak, saves the satellite state at the start of the streak, then orders all the streak data in time-increasing order. Finally, the streak generation program alters the accurate outputs by adding normally-distributed errors. This step imitates the impact of inaccurate orbit determination that results from short streaks. The size of the perturbations was varied from no variation to variations whose standard deviation is of the order of five percent of selected parameters. These largest variations are representative of those arising from realistic orbit determination applied to short streaks.

Because of the complexity of the problem of dealing with the 10,000 satellites in a comprehensive catalog, the program was run for several small catalogs of 10, 47, 107, and 182 satellites. Of these satellites, only 6, 36, 74, and 137 satellites, respectively, created streaks. These counts, then, are the correct number of clusters we desired from the clustering processes. As a matter of information, the runs generated 95, 861, 2191, and 4408 streaks, respectively. Although small compared to the 200,000 streaks that might be collected during a year, these sample cases provided sufficient size and complexity to test the different clustering algorithms we considered.

The next step in the test bed was to exercise the algorithms we had chosen. In a perfect setting, the number of clusters should equal the number of satellites making streaks. However, errors in the data used for clustering will undoubtedly cause the clustering algorithm to estimate a number of clusters different from the true count. In addition, the clustering algorithm might misassign streaks to the wrong satellites. These two errors provide some insight into the accuracy of the clustering process over and above the strictly statistical measures normally employed.

Because of time constraints, we used MATLAB algorithms rather than write our own code. We recognize that an operational system will require specialized algorithms and code that addresses the size of the problem, which might dictate parallel processing and other computational throughput devices. Each of the algorithms provided by MATLAB permits the user to select the weighting and linkage. The following report sections discuss a number of factors associated with the clustering tests we performed with the streak data supplied by Dr. Green - computational complexity, cluster accuracy as measured by comparing the number of clusters with the correct number of satellites and accuracy of streak assignments to clusters.

4.1 Clustering

As a result of theory we were aware of the computational expense we were facing by using the hierarchical method. Hence before we decided to commit to the size of the experimental data we would work with, we decided to test what are our computational limitations were. Table 1 was constructed based of our test runs with unperturbed data.

In Table ??we learn about how much time it will take to cluster a certain number of streaks that were generated by a corresponding number of satellites. We performed the calculations for the standard Kepler coordinates and the Cartesian coordinates of an ellipse. After running this experiment we see that the time it takes to run the clustering algorithm growth very quickly as the number of streaks increases. Based of the test run

Satellites	Streaks	Kepler Time	Ellipse Time
6	96	.05	.06
32	861	3.85	4.48
74	2191	56.45	61.7
137	4086	423.13	443.17

Table 1: Computational time (seconds)

we decided to limit ourselves to about five minutes of running time. Running this test experiment also tested that our code was working for unperturbed data and hence we were able to turn our attention to the perturbed cases.

Since we are using the inbuilt Matlab function for clustering we are limited to the distance and linkage functions provided by Matlab. In order to understand better which distance function to use, we ran an experiment testing which routine will work the best. In Table 2 we test the euclidean, weighted and cosine distance routines.

Satellites	Euclidean	Weighted	Cosine
6	63	7	644
36	612	99	617
74	1563	273	1537
137	3107	764	3098

Table 2: Performance of norms (# clusters)

The decision for which distance function is the best choice is done by comparing the number of clusters the routine produces to the number of satellites making streaks. From the results it is clear that the weighted distance function is the best choice. The next step is to understand how the linkage affects the clustering output. In Table 2 we used the single linkage routine for all the experiments.

The next experiment we used the weighted distance but changed the linkage function between single, average, and centroid. The results are listed in Table 3.

Satellites	Single	Average	Centroid
6	7	13	13
36	99	86	82
74	273	260	240
137	764	520	472

Table 3: Performance of linkage (# clusters)

Again we compared the number of clusters generated by the method to number of satellites. There are slight improvements from using average and centroid linkage functions

but not enough to make a confident conclusion. The final parameter that we have control over in matlab routine is the cut-off. The cut-off parameter corresponds to the threshold of the clustering routine. Table 4 shows how the cut-off affects the clustering result. In this experiment, the clustering routine runs with weighted distance function and the single linkage function. The results that we get provide great insight, because we see how sensitive the cut-off value is.

Satellites	Found	Cut-off	Silhouette
6	6	1.154	0.70
36	32	1.1546	0.70
36	33	1.1547	0.79
74	57	1.1546331	0.48
137	133	1.1546	0.47

Table 4: Effect of cut-off on silhouette (a, e weighted with 0.1)

The silhouette parameter measures the quality of the assignment process. A value of 1 represents a perfect assignment and the value of -1 represents a bad assignment. The magnitude of the silhouette values and their relative invariant sizes indicate inconclusive results with poor assignment. Based of the experiment from Table(4) we conclude that there needs to be more theoretical work done for choosing the cut-off parameter intelligently.

4.2 Sectioning Method

We originally stated that the cardinality of this problem can be problematic because clustering algorithms are computationally expensive. We proposed the sectioning method that may cut the computation time of clustering large amounts of data. The idea of the method is to break down the starting set into smaller subsets, and perform the clustering method on each subset. After each subset is clustered we calculate the centroids of each cluster in all the subsets and we cluster the centroids. We experimented the idea with the data that had 4406 streaks. The results of the experiment are stated in Table 5. These

Sections	1	2	4	8
Time	356	143	56	12
Found	137	116	126	143

Table 5: Effective grouping

results especially for grouping into 8 and 4 subgroups looks very promising as far as the number of clusters is concerned. This idea needs to be taken further by figuring out what is the optimal number of subsets the original set needs to be broken into.

4.3 Varying the value of cutoff in hierarchical clustering

The *clusterdata* function in Matlab is an implementation of the hierarchical clustering algorithm, one of the parameters the function takes is the *cutoff*. This parameter is a threshold for cutting the hierarchical tree generated by a linkage function when making clusters. For purposes of experimentation with the data we had, we varied the value of the cutoff until we could get as close as possible to the true value of the number of clusters that were in the data supplied. We realize that for practical purposes this is not the way to solve the clustering problem, but it gives us an idea of the range of cutoff that should be used for a particular data set. The table below shows the results obtained using a weighted euclidean norm on the first 5 components of the vectors from the parameter space.

Table 6: Effect of cutoff, weights=[0.09,0.1,1,1,1]

Number of streaks	Num of Satellites found	Actual Number of Satellites	Cutoff
95	6	6	1.154
861	32	36	1.1546
2191	57	74	1.15463
4408	133	137	1.1546

The weights used in Table6 are obtained brute force taking into account the relative importance of each of the parameters in describing the orbit. It is clear that this does not provide the best cluster analysis, better results can be obtained by a careful analysis of the equations governing the motion of the objects in the orbit described by the elements of the parameter space.

4.4 *k-means* Clustering

The *k-means* clustering algorithm has also been considered in solving the clustering problem. However success of the *k-means* algorithm hinges on knowing the value of *k*, the number of clusters before clustering is done. Due to the large number of observations that are anticipated with improving technology, it is almost impossible to determine the number of clusters before hand. One approach we tried with small data sets to work around this problem was to try different values of *k* and use the matlab *silhouette* function to measure the quality of the clustering. A drawback to this approach is that as the number of clusters increases on some iterations the algorithm terminates as it creates empty clusters which are meaningless for our classification.

5 Conclusion – Future Work

By studying three parameter spaces, a few similarity functions have been defined in order to be able to implement an agglomerative hierarchical clustering algorithm. Given that the algorithm will be applied to a large set of data, a low-memory adaptation has been studied.

Acknowledgements

We would like to thank the Institute for Mathematics and its Applications and The Aerospace Corporation for making this workshop possible.

References

- [1] O. Duda, Richard and E. Hart, Peter. Use of the Hough transformation to detect lines and curves in pictures. *Technical Note 36*, 1971.
- [2] Levent Ertoz, Michael Steinbach, and Vipin Kumar. *A New Shared Nearest Neighbor Clustering Algorithm and its Applications*. Workshop on Clustering High Dimensional Data and its Applications at 2nd SIAM International Conference on Data Mining. 2002.
- [3] Guojun Gan, Chaoqun Ma, and Jianhong Wu. *Data Clustering: Theory, Algorithms, and Applications*. ASA-SIAM Series on Statistics and Applied Probability. 2007.
- [4] J.C. Gower and P. Legendre. Metric and euclidean properties of dissimilarity coefficients. *Journal of Classification*, 2005.
- [5] Leonard Kaufman and J. Rousseeuw, Peter. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley-Interscience, 1990.
- [6] Jacob Kogan, Charles Nicholas, and Marc Teboulle. Clustering large and high dimensional data. 2003.
- [7] David Littau and Daniel Boley. Using low-memory approximations to cluster very large data sets.
- [8] W. Milligan, Glenn and C. Cooper, Martha. A study of standardization of variables in cluster analysis. *Journal of Classification*, 2005.
- [9] Dan Pelleg and Andrew Moore. *X-means: Extending K-means with Efficient Estimation of the Number of Clusters*. Proceedings 17th International Conference on Machine Learning. 2000.

- [10] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison Wesley, 2005.
- [11] A. Vallado, David. *Fundamentals of Astrodynamics and Applications*. second edition, 2004.
- [12] http://en.wikipedia.org/wiki/right_ascension.