

ORBIT CLUSTERING BASED ON TRANSFER COST

Eric D. Gustafson*, Juan J. Arrieta-Camacho† and
Anastassios E. Petropoulos‡

We propose using cluster analysis to perform quick screening for combinatorial global optimization problems. The key missing component currently preventing cluster analysis from use in this context is the lack of a useable metric function that defines the cost to transfer between two orbits. We study several proposed metrics and clustering algorithms, including k -means and the expectation maximization algorithm. We also show that proven heuristic methods such as the Q-law can be modified to work with cluster analysis.

INTRODUCTION

A common challenging problem for mission designers is to design a tour of multiple bodies chosen from a given set. The global combinatorial nature of the optimization problem almost certainly precludes an exhaustive solution for a large enough set of bodies. Typically, a screening method is used to select a subset of bodies that can then be studied in more detail. This screening process is critical to finding a satisfactory solution. In this paper, we propose using cluster analysis as a fast and effective method to perform this initial screening given a set of closed orbits ($0 \leq e < 1$) in the restricted two-body problem.

The motivation for this work was the 5th Global Trajectory Optimization Competition (GTOC).^{*} At a very high level, the problem posed by the 5th GTOC involved visiting a sequence of asteroids with a low-thrust spacecraft. The asteroids were to be chosen from a list of 7,075 possible asteroids, representing just a small sample of the many small bodies in our solar system.

There are many clustering algorithms available for partitioning data. The common foundation among all clustering methods is the existence of a metric function that defines the “distance” between two members of the set. In the case of orbital transfers, this metric could be any parameter of interest such as optimal delta- V or time of flight.

Unfortunately, when assessing large sets of orbits, computing the optimal transfer cost between *all* pairs of orbits would be far too time consuming. An additional complication is that some heuristic methods do not provide a mathematically valid metric, which makes them incompatible with most clustering methods. For instance, many successful methods such as the Q-law¹ are not symmetric and do not obey the triangle inequality.

*Navigation Engineer, Mission Design and Navigation Section, Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, California 91109.

†Navigation Engineer, Mission Design and Navigation Section, Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, California 91109.

‡Senior Engineer, Mission Design and Navigation Section, Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, California 91109.

¹<http://www.esa.int/gsp/ACT/mad/pp/GTOC/index.htm>

These problems raise a few interesting questions: What makes a good metric function for orbital transfers? What is a good parameterization of the orbits? Which clustering algorithms work well?

To answer these questions, we perform a study of various metric functions, parameterizations, and clustering methods. The clustering algorithms to be studied are standard, well-known algorithms: k -means,² fuzzy c -means (FCM),³ expectation maximization (EM),⁴ and a graph partitioning method. The k -means algorithm and its variants partition set members into discrete clusters such that each member of the cluster is closer to the cluster's mean than any other cluster's mean. In contrast to the discrete clusters computed by k -means, the fuzzy c -means and EM algorithms assign to each set member a probability that it is a member of a given cluster. This has the advantage of not only providing groupings of orbits, but also a nearest-neighbor type of ranking.

We will study two metrics – one a modification to the well-established Q-law, and the other new. We show that even though the Q-law does not produce symmetric costs, nor does it obey the triangle inequality, it is trivial to form a symmetric Q-law variation, and that the triangle inequality violation does not necessary cause clustering algorithms to fail. The new metric is based on a different parameterization of the orbit size and shape. We show that the angular momentum vector and eccentricity vectors, scaled by factors to account for the ease by which the orbit may be change by an applied acceleration, provide a vector space in which the standard Cartesian norm can be used as the distance metric.

CLUSTERING BACKGROUND

Clustering is the process of assigning members of a data set to a number of partitions such that the elements of the partition are close to each other in some sense. Ideally, the clusters themselves are space far apart from each other. Cluster has found many applications in biology, data mining, computer science and other varied fields. We will briefly describe the the four clustering algorithms mentioned above, showing example results on a simulated two-dimensional data set shown in Fig. 1. The simulated data is an equally-weighted Gaussian mixture of five distributions, creating five clusters that are obvious to the human eye, but not always so obvious to clustering algorithms.

k -means Algorithm

Consider the problem of partitioning the dataset

$$X = \{1, 5, 7, 9, 12, 13\}. \quad (1)$$

into $K = 3$ disjoint, non-empty subsets c_1 , c_2 , and c_3 whose union is X . It would not be unreasonable to suggest

$$c_1 = \{1, 5\}, \quad c_2 = \{7, 9\}, \quad \text{and} \quad c_3 = \{12, 13\}. \quad (2)$$

In measuring how appropriate is such partition, one could compute the arithmetic mean for every cluster subset:

$$\mu_1 = \frac{1+5}{2} = 3, \quad \mu_2 = 8, \quad \text{and} \quad \mu_3 = 12.5, \quad (3)$$

and calculate the *squared error* between μ_k and the points in c_k from

$$J(c_k) = \sum_{x_i \in c_k} ||x_i - \mu_k||^2, \quad k = 1, 2, 3 \quad (4)$$

to yield

$$J(c_1) = (1 - 3)^2 + (5 - 3)^2 = 8, \quad J(c_2) = 2, \quad \text{and} \quad J(c_3) = 0.5, \quad (5)$$

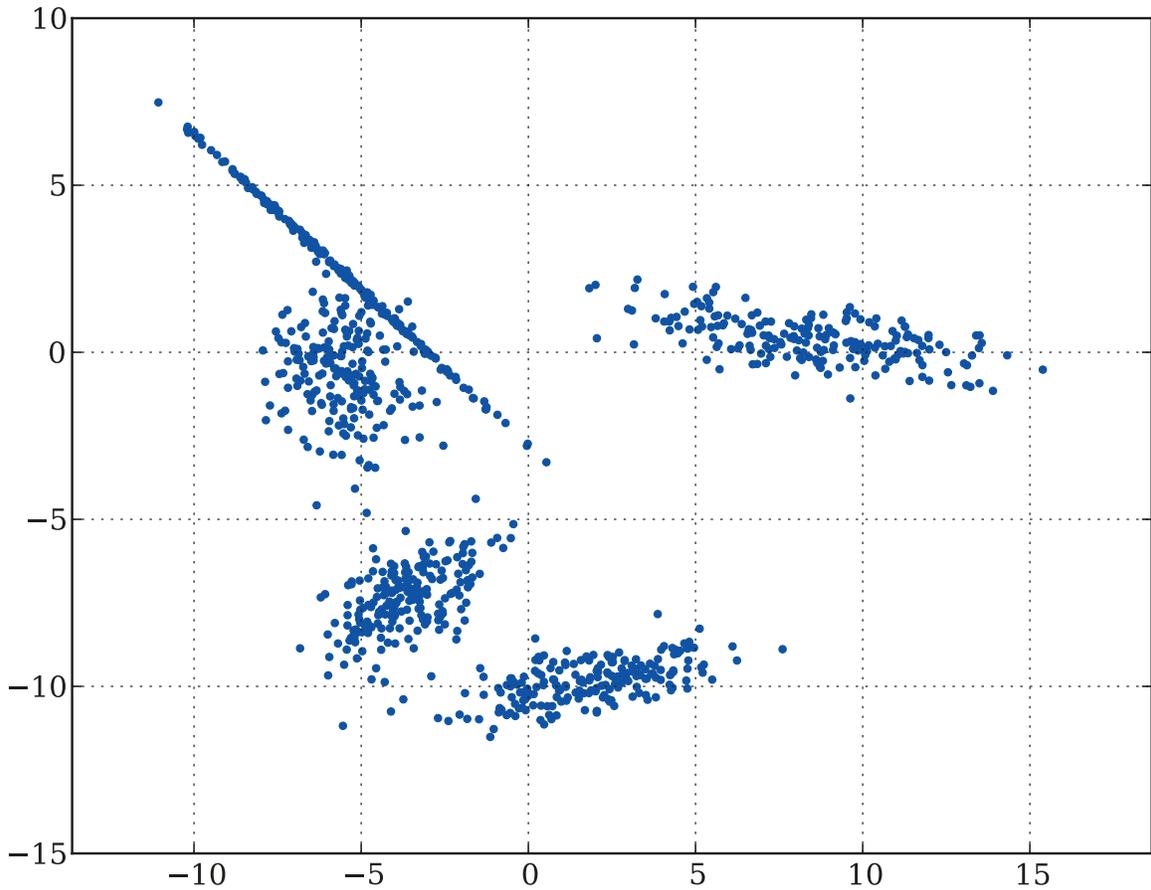


Figure 1. Randomly generated 2-D data set.

for a total among subsets of

$$J(C) = \sum_{k=1}^3 J(c_k) = 8 + 2 + 0.5 = 10.5. \quad (6)$$

As it turns out, it is possible to select a different dataset partitioning that would result in a lower $J(C)$. Indeed, selecting

$$c_1 = \{1\}, \quad c_2 = \{5, 7, 9\}, \quad \text{and} \quad c_3 = \{12, 13\} \quad (7)$$

yields $J(C) = 8.5$ —the absolute minimum value of J for the given dataset and three partitions.

The k -means algorithm is a generalization of the previous procedure for an arbitrary X and K ; its goal is to generate the partition that minimizes the sum of the squared error (4) over all K subsets (known as *clusters*) in the following manner:

1. Select an initial partition with K clusters; repeat 2 and 3 until cluster membership stabilizes.
2. Generate a new partition by assigning each item to its closest cluster center.

3. Compute new cluster centers.

Even the small, one-dimensional example presented above highlights some of the challenges inherent to this algorithm. For example, the initial selection of the number and location of clusters is mostly arbitrary, and a poor selection of either can lead to poor run-time performance, clustering quality, or both.

In addition, being defined over a metric space, the algorithm requires a measure of *distance*, which can be challenging to define for points representing entities different from positions in the Euclidean space.

Finally, the global minimization of (4) is a known NP-hard problem, which forces the algorithm to provide local minima for all but the simplest of problems. Despite its shortcomings, *k*-means clustering remains one of the most popular and simple clustering algorithms, and it can provide rapid insights into the groupings of datasets.

Since the initial choice of centers directly impacts the final clustering, much work has been spent studying methods to improve the initialization of the *k*-means algorithm. An algorithm called *k*-means++ is meant to choose cluster centers in a way such that the standard *k*-means algorithm is much more likely to run faster and give better results.⁵ In this paper, we always use *k*-means++ to initialize our *k*-means runs. Defining the set of points to be clustered as X and $D(x)$ as the distance between any point and its nearest center, the *k*-means++ algorithm can be written as follows:

1. Chose one center at random from X
2. Chose the next center from X with probability $\frac{D^2(x)}{\sum_{x \in X} D^2(x)}$
3. Repeat Step 2 until k centers are chosen.
4. Proceed with the *k*-means algorithm, initialized with the centers from above.

Figure 2 shows the results of a *k*-means clustering on the example data set. Each cluster is drawn in a different color, outlined by the convex hull. The diamonds mark the initial cluster centroids determined by *k*-means++, and the squares mark the final. Clearly *k*-means does a very good job on the clusters with large inter-cluster separation.

Fuzzy *c*-means Algorithm

Instead of the hard clustering of *k*-means, fuzzy *c*-means assigns to each particle a cluster weight for each cluster. This can be represented by a matrix W with elements w_{ij} , where i is the point index and j is the cluster index. Each $w_{ij} \in [0, 1]$, where a value of zero means the point i has no belonging in cluster j , and a value of one means the point i completely belongs to cluster j . Additionally, $\sum_j w_{ij} = 1$, so the weighting values may be viewed as probabilistic cluster assignments, hence the “fuzziness.”

Given in initial weight matrix, and a user-specified value for the parameter m , the fuzzy *c*-means algorithm is as follows:

1. Choose initial weighting assignment matrix, W . We used *k*-means for this, therefore w_{ij} equals one or zero.

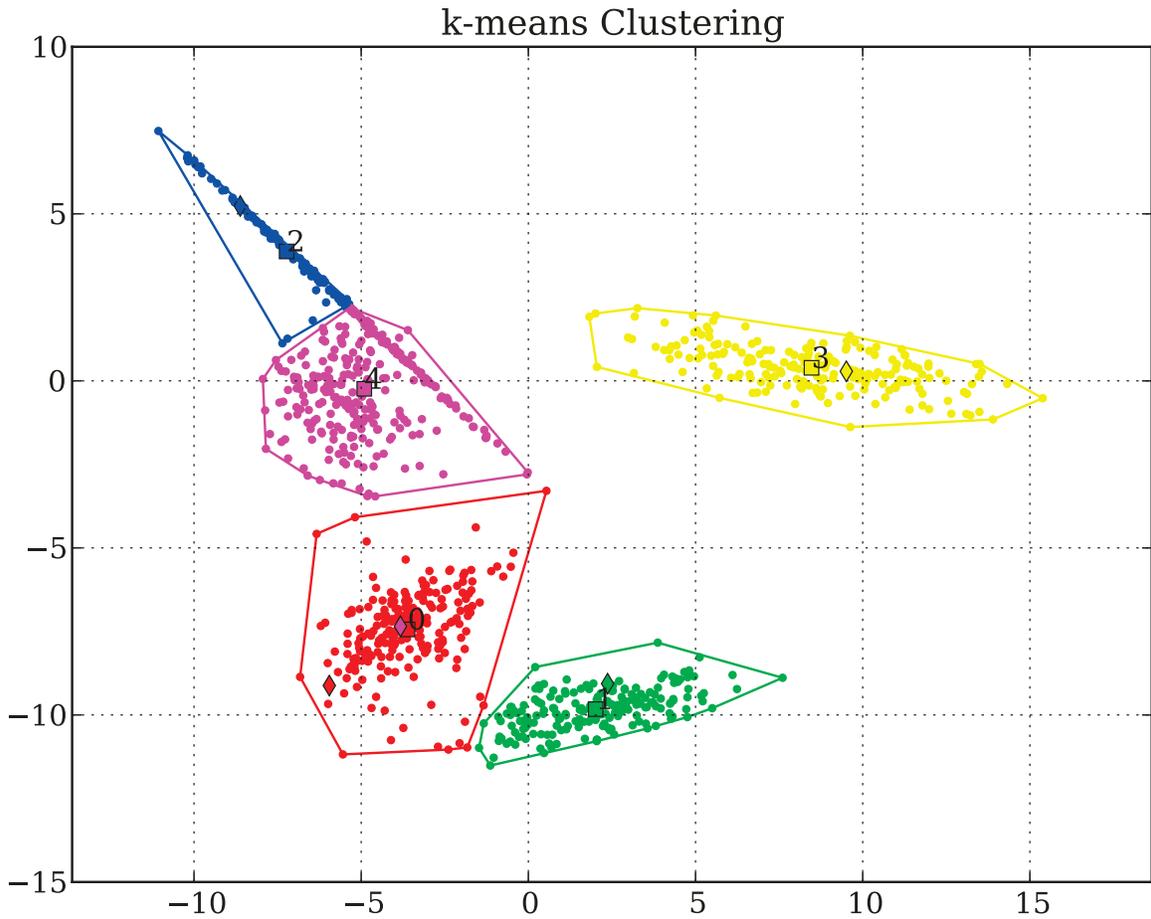


Figure 2. Example of k -means clustering on example 2-D data set.

2. Compute the centers using

$$c_k = \frac{\sum_i w_{ik} x_i}{\sum_i w_{ik}}, \quad \text{where } x_i \text{ is the point } i \quad (8)$$

3. Compute the new weights using

$$w_{ik} = \left(\sum_j \left(\frac{d(c_k, x_i)}{d(c_j, x_i)} \right)^{2/(m-1)} \right)^{-1} \quad (9)$$

4. Repeat Steps 2 and 3 until the maximum weight change is no larger than some tolerance.

Figure 3 shows the results of a fuzzy c -means clustering on the example data set. Each cluster is drawn in a different color, outlined by the convex hull of points in a cluster with weighting greater than 0.8. The clustering is similar to k -means, but as seen, the boundaries of the clusters at a given threshold can overlap.

Figure 3. Example of fuzzy c -means clustering on example 2-D data set.

Expectation Maximization Algorithm

The expectation maximization algorithm is another soft clustering method like fuzzy c -means, but with a more mathematically formal basis and well-established convergence behavior. In particular, expectation maximization is perfectly suited to estimating the parameters of Gaussian mixtures (GM). We will abbreviate this pair as EM-GM. EM-GM produces a weight matrix, called γ_{ij} here, along with a cluster weight, mean and covariance : w_j , μ_j and Σ_j . Let the data points be y_i , $i = 1, \dots, n$, be clustered into k groups. Then the EM-GM algorithm is as follows:⁶

1. Choose initial parameters for $w_j^{(0)}$, $\mu_j^{(0)}$ and $\Sigma_j^{(0)}$. As with fuzzy c -means, we used k -means to perform this initialization.
2. Expectation step: For each cluster ($j = 1, \dots, k$), compute the new weight matrix:

$$\gamma_{ij}^m = \frac{w_j^{(m)} \phi(y_i | \mu_j^{(m)}, \Sigma_j^{(m)})}{\sum_{l=1}^k w_l^{(m)} \phi(y_i | \mu_l^{(m)}, \Sigma_l^{(m)})}, \quad i = 1, \dots, n \quad (10)$$

and

$$n_j^{(m)} = \sum_{i=1}^n \gamma_{ij}^{(m)}, \quad (11)$$

where $\phi(y_i | \mu_j^{(m)}, \Sigma_j^{(m)})$ is the value of the Gaussian probability distribution function (pdf) with mean μ_j and covariance Σ_j evaluated at y_i during the m -th iteration.

3. Maximization step: For each cluster ($j = 1, \dots, k$), compute the new pdf parameters:

$$w_j^{(m+1)} = \frac{n_j^{(m)}}{n}, \quad (12)$$

$$\mu_j^{(m+1)} = \frac{1}{n_j^{(m)}} \sum_{i=1}^n \gamma_{ij}^{(m)} y_i, \quad (13)$$

$$\Sigma_j^{(m+1)} = \frac{1}{n_j^{(m)}} \sum_{i=1}^n \gamma_{ij}^{(m)} (y_i - \mu_j^{(m+1)}) (y_i - \mu_j^{(m+1)})^T. \quad (14)$$

4. Convergence check and iteration: Compute the new log-likelihood:

$$\ell^{(m+1)} = \frac{1}{n} \sum_{i=1}^n \ln \left(\sum_{j=1}^k w_j^{(m+1)} \phi(y_i | \mu_j^{(m+1)}, \Sigma_j^{(m+1)}) \right) \quad (15)$$

If $|\ell^{(m+1)} - \ell^{(m)}| > \varepsilon$ for some tolerance ε , then return to step 2.

Figure 4 shows the results of an EM-GM clustering on the example data set. The clusters in the case are quite different from k -means and fuzzy c -means. For each cluster, the 1-, 2-, 3-, and 6-sigma ellipses are drawn with successively more transparency. EM-GM perfectly found the long skinny distribution in cluster 2, although one could argue this is expected since the underlying data was generated with a Gaussian mixture—exactly what EM-GM is designed to find.

Graph Partitioning Algorithm

Clustering algorithms may also be applied to graphs, where the distance between each elements is no longer important; the only necessary inputs are the connections between the elements. Clustering is performed by partitioning the graph based on the edge structure, where ideally there are many edges within each cluster and few edges between clusters.

The question then is how to compute the connections given a set of points. Conceptually, one could construct an n -nearest neighbor graph, however, this is computationally expensive and once the neighbors and their distances are computed, the value of clustering is possibly diminished. In this study, we used Delaunay triangulation, implemented using the Qhull program.⁷ The underlying idea is that the Delaunay triangulation tends to put many edge connections in regions of high density, and few connections where points are sparse. See Fig. 5 for the Delaunay triangulation of the example data set. Notice the large number of edges in the dense regions, and fewer edges in regions that are clearly between clusters. One could further improve the graph, for example, by removing the longest edges, however, we simply used the edges directly from Qhull.

The second part of the algorithm is to partition the graph. We used the multi-level partitioning code METIS⁸ to do this. We found METIS to be extremely fast as discussed later. Figure 6 shows

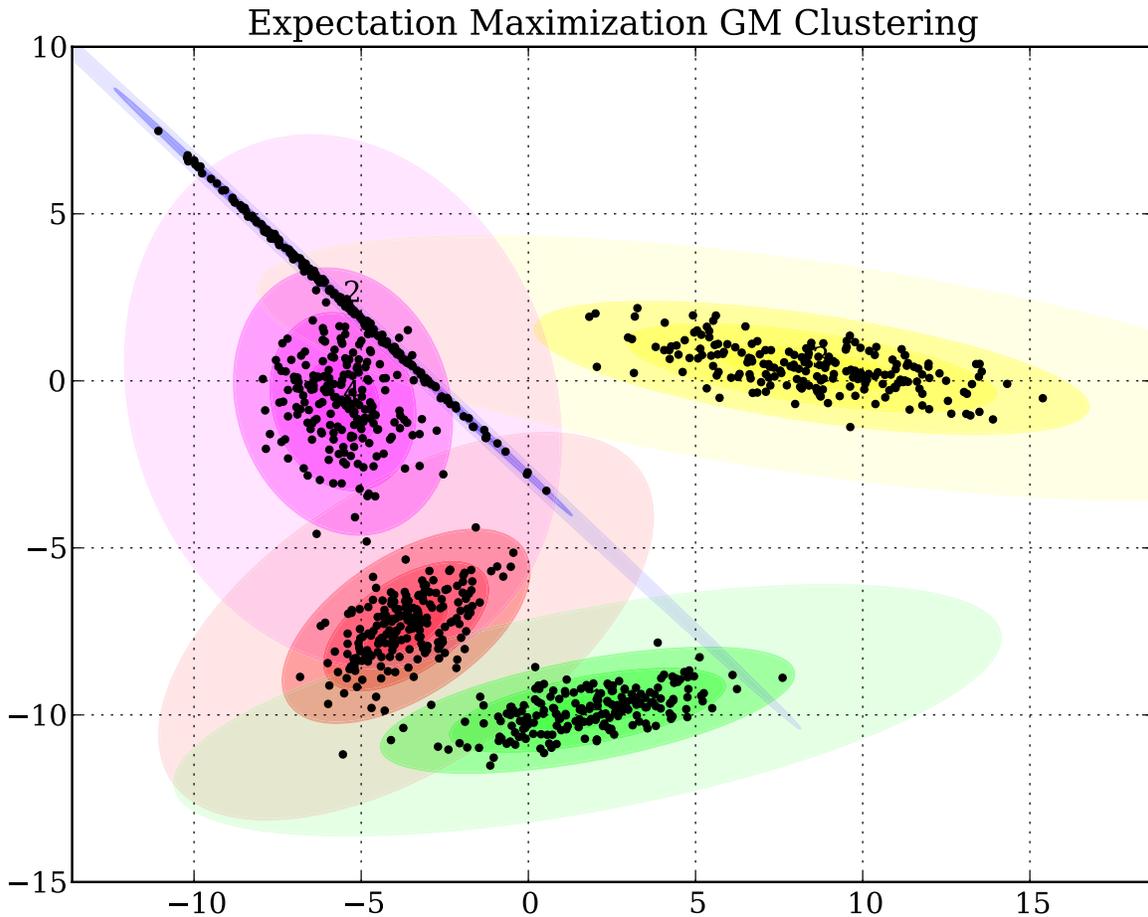


Figure 4. Example of EM-GM clustering on example 2-D data set.

the clustering result for the example data set. The cluster assignments are not too different from the fuzzy c -means method, even though the clustering approach is totally different.

DISTANCE METRICS

A distance metric is simply a function defining a distance between two elements of a set. We use the term distance loosely. In this application, the set elements are Keplerian orbits, and the distance between them is some approximation to the optimal cost of transferring between. Note that for many cost functions, the optimal transfer cost between a pair of orbits satisfies the criteria for a metric function. Regarding the approximation for clustering purposes, the distance metric doesn't need to be particularly accurate for orbits far away from each other. As long as the metric gives an answer that is large enough relative to the intra-cluster distances, the clustering result will not be affected.

All metric functions must satisfy the four conditions in Table 1, where $d(x, y)$ is a metric function, and x and y are some parameterization of two orbits.

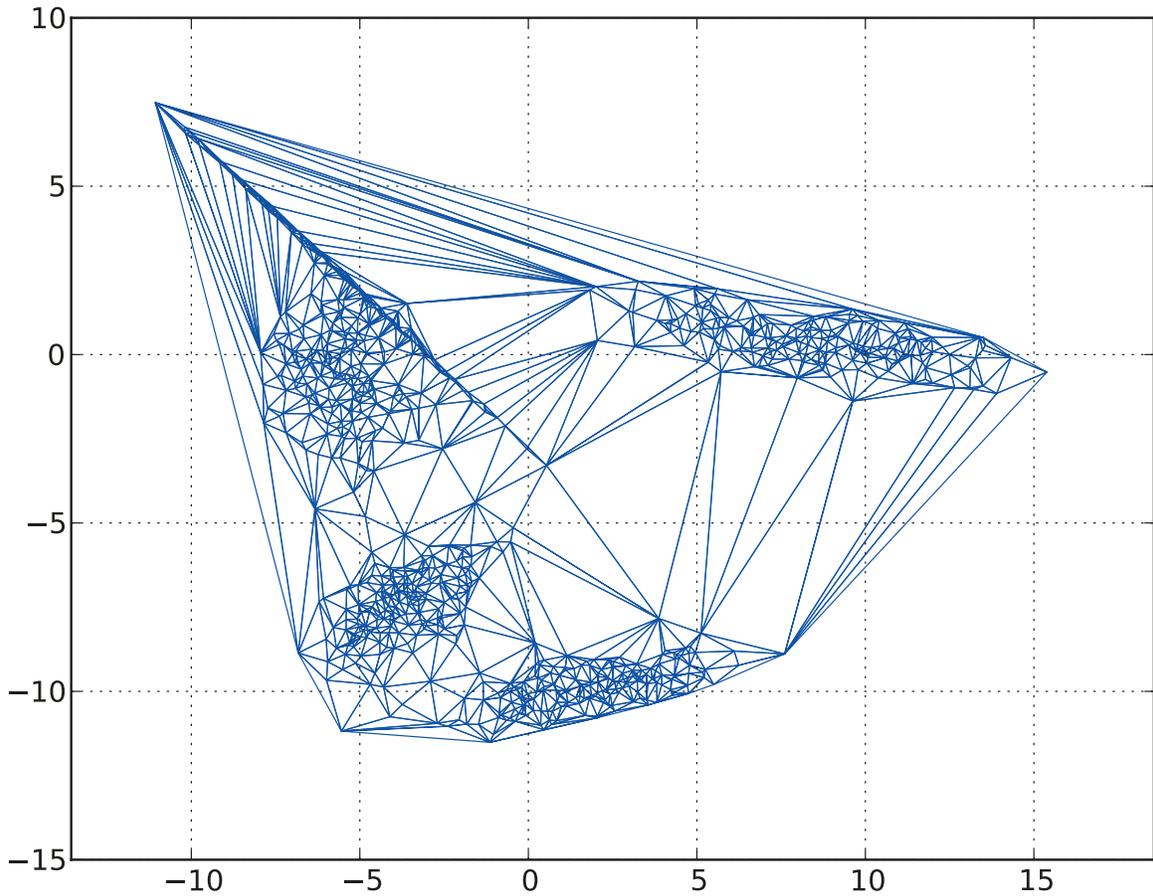


Figure 5. Delaunay triangulation of example 2-D data set.

Table 1. Distance Metric Criteria

Criterion	Description
$d(x, y) \geq 0$	non-negativity
$d(x, y) = 0$ if and only if $x = y$	“Identity of indiscernibles”
$d(x, y) = d(y, x)$	symmetry
$d(x, z) \leq d(x, y) + d(y, z)$	triangle inequality

Q-law

In the Q-law,¹ the proximity quotient, Q , is defined, serving as a candidate Lyapunov function for use in Lyapunov feedback control for transfers between orbits using low-thrust. In summary, the proximity quotient attempts to judiciously quantify the proximity of the osculating orbit to the target orbit. It may be thought of as the best-case quadratic time-to-go. Q is defined as follows:

$$Q = (1 + W_p P) \sum_{\alpha} W_{\alpha} S_{\alpha} \left[\frac{d(\alpha, \alpha_T)}{\tilde{a}_{xx}} \right]^2, \quad \text{for } \alpha = a, e, i, \omega, \Omega \quad (16)$$

where the five orbital elements (α) are the semimajor axis (a), eccentricity (e), inclination (i), argument of periapsis (ω), and longitude of the ascending node (Ω); W_p and the W_{α} are scalar weights

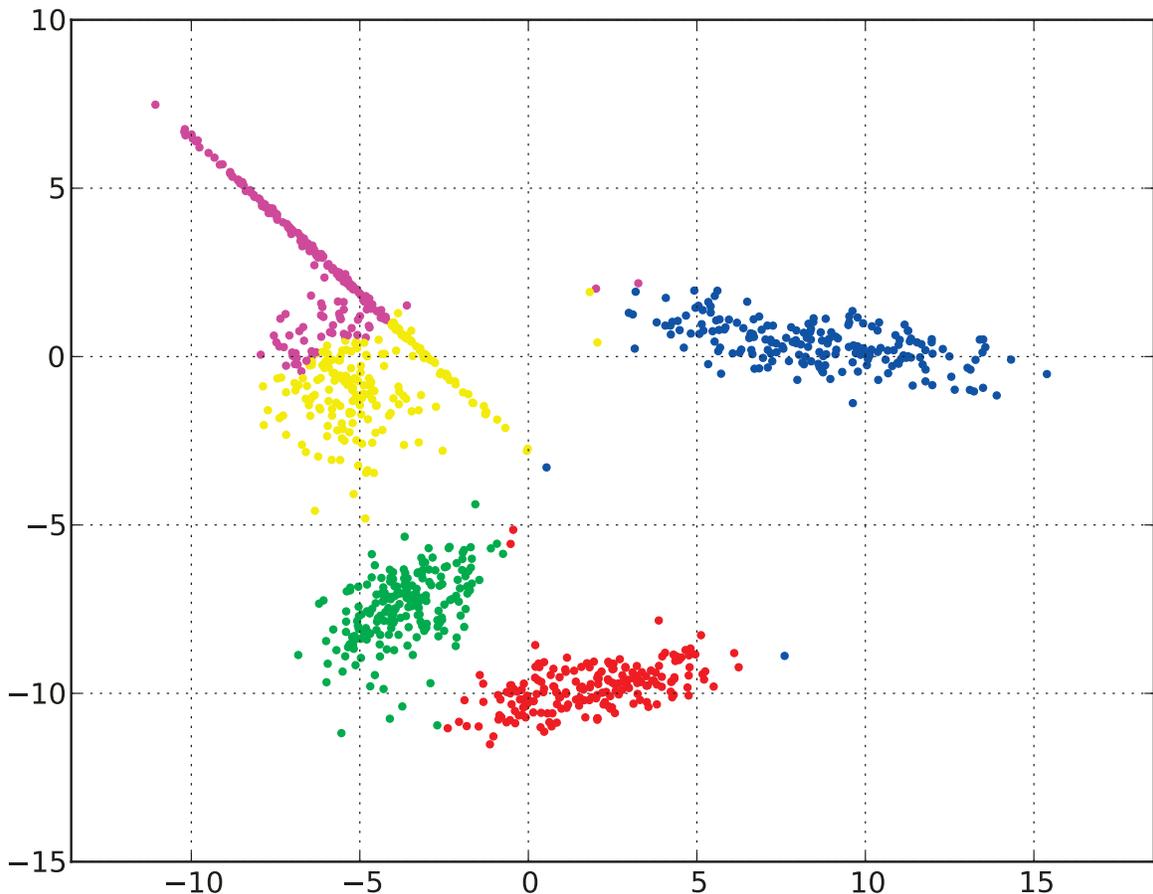


Figure 6. Example of graph clustering on example 2-D data set.

greater than or equal to zero; the subscript T denotes the target orbit element value (without subscript, the osculating value is indicated); $\tilde{\alpha}_{xx}$ nominally denotes the maximum over thrust direction and over true anomaly on the osculating orbit of the rate of change of the orbit element (due to thrust); P is a penalty function; S_{α} is a scaling function; and $d(\alpha, \alpha_T)$ is a distance function:

$$d(\alpha, \alpha_T) = \begin{cases} \alpha - \alpha_T & \text{for } \alpha = a, e, i \\ \cos^{-1}[\cos(\alpha - \alpha_T)] & \text{for } \alpha = \omega, \Omega \end{cases} \quad (17)$$

where the principal value, namely $[0, \pi]$, is used for the arc cosine. The peculiar form of the distance function for ω and Ω is used because it provides an angular measure of the distance between two positions on a circle using the “short way round” the circle, because it is differentiable with respect to α [except when $d(\alpha, \alpha_T) = \pi$], and because the sign of the derivative indicates whether α leads or lags α_T based on the short way round.

The proximity quotient, Q , is a very useful heuristic in defining the ease of transfer between two orbits, however, it only adheres to the first two conditions in Table 1. In this section, we will describe methods by which a distance metric can be transformed to overcome its violations of symmetry and the triangle inequality. First, any asymmetric pseudo-metric can be used to form a symmetric metric

by forming an appropriate function combining the two bilinear forms. Let an asymmetric pseudo-metric be $\tilde{d}(x, y)$. In this study, we enforced symmetry of the proximity quotient Q by defining a symmetric metric as

$$d'(x, y) = \min(\tilde{d}(x, y), \tilde{d}(y, x)). \quad (18)$$

Given a set of elements $\{x_1, x_2, \dots, x_n\}$, one may compute the pseudo-metric between each pair of elements, and store the values in a matrix \tilde{D} such that $\tilde{D}_{ij} = \tilde{d}(x_i, x_j) = \tilde{d}_{ij}$. Applying Eq. 18 to D gives $D' = \min(\tilde{D}, \tilde{D}^T)$, where $\min(\cdot)$ is the element-wise minimum. Another interesting choice, not studied here, would be to use the p -norm instead of the minimum:

$$d'(x, y) = (\tilde{d}(x, y)^p + \tilde{d}(y, x)^p)^{\frac{1}{p}}, \quad p \geq 1$$

A rather naïve but effective algorithm to ensure the triangle inequality is shown is Algorithm 1. To use the algorithm, one would repeatedly call function UPDATED until the output is identical to the input. The algorithm simply searches for a shorter path between elements i and j through element k :

$$\tilde{d}_{ij} = \min_k(d_{ik} + d_{kj}).$$

Although convergence behavior was not formally analyzed, several tests on matrices up to $7,075 \times 7,075$ never required more than 6 iterations for the algorithm to converge.

Algorithm 1 Triangle Inequality Enforcement

Require: \tilde{D}' is symmetric and size $n \times n$

```

1: function UPDATED( $D'$ )
2:    $V \leftarrow 0_{n \times n}$  ▷ Zero matrix of size  $n \times n$ 
3:   for  $i \leftarrow 1, n$  do
4:      $\vec{v}_1 \leftarrow D'_{i:}$  ▷ The  $i$ -th row, transposed to a column
5:     for  $j \leftarrow i + 1, n$  do
6:        $\vec{v}_2 \leftarrow D'_{:j}$  ▷ The  $j$ -th column
7:        $\vec{v}_k \leftarrow \vec{v}_1 + \vec{v}_2$ 
8:        $v \leftarrow \min(\vec{v}_k)$ 
9:        $V_{ij} \leftarrow v$ 
10:       $V_{ji} \leftarrow v$ 
11:    end for
12:  end for
13:  return  $V$ 
14: end function

```

The present transformation of the proximity quotient into a quantity that satisfies the distance-metric criteria of Table 1 can affect a sizeable portion of asteroid orbit pairs. We can quantify the impact by computing the ratio of the modified proximity quotient to the original for every pair of orbits. Figure 7 shows a histogram of the cost ratios for all orbit pairs in the GTOC-5 set. For many pairs, the ratio is exactly one, indicating that the original proximity quotient could not be improved upon. However, for most pairs, the ratio is less than one, which means the transformation to enforce symmetry and the triangle inequality resulted in a smaller value of the distance metric.

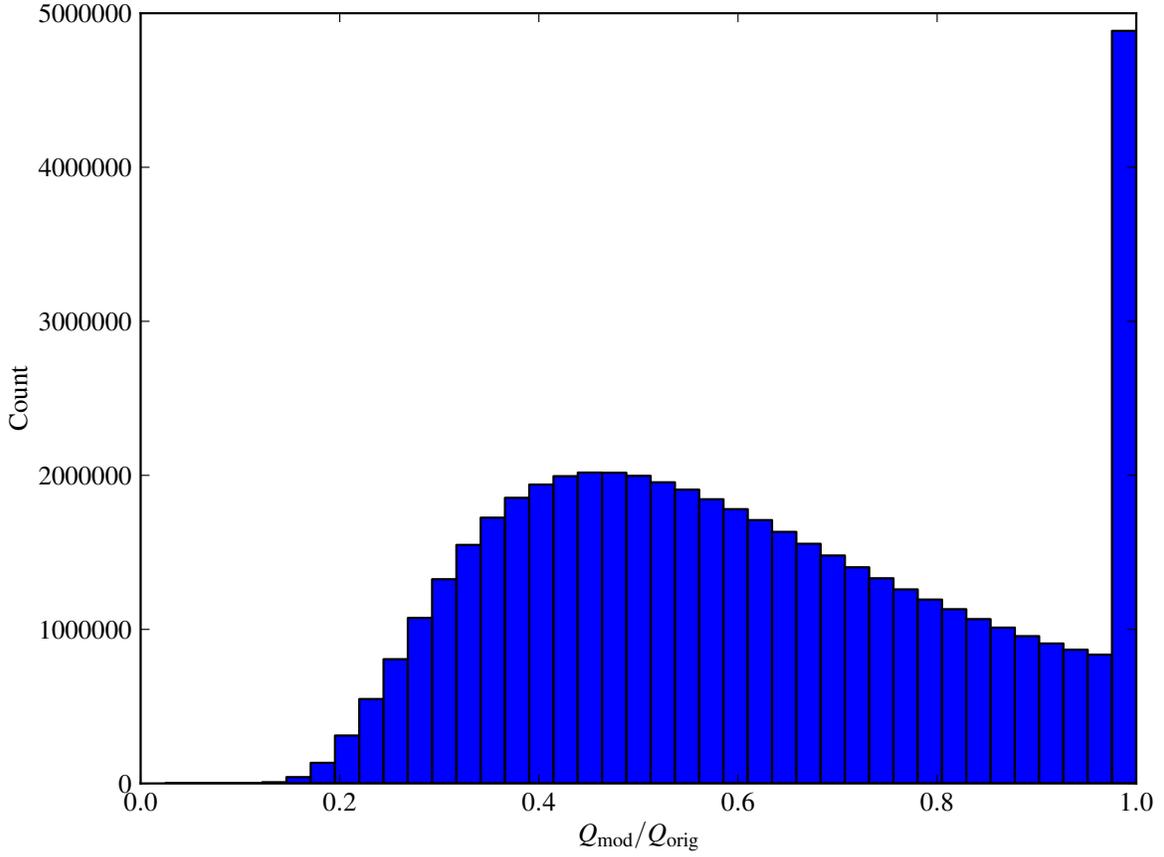


Figure 7. Modified proximity quotient, Q_{mod} , as compared to the original.

A subtlety of using centroid-based clustering algorithms such as k -means and fuzzy c -means with the Q-law is the computation of cluster centroids. These methods assume the objects being clustered are vectors, which are trivial to average. When using the Q-law, the underlying representation of orbits is a set of Keplerian elements, which do not form a vector space. Therefore, we need a more general solution, which can be obtained by realizing that the vector mean satisfies an optimization problem formulated in a general sense. Given n vectors, \vec{r}_i , the mean, \vec{r}_G , can be computed directly:

$$\vec{r}_G = \frac{1}{n} \sum_i \vec{r}_i. \quad (19)$$

However, we can also define \vec{r}_G indirectly as follows. Define $\vec{\rho}_i$ as the relative position of \vec{r}_i with respect to \vec{r}_G : $\vec{\rho}_i = \vec{r}_i - \vec{r}_G$. It is straightforward to show that the cost function

$$J = \sum_i \vec{\rho}_i \cdot \vec{\rho}_i \quad (20)$$

is minimized by the solution in Eq. (19). The cost function in Eq. (20) is the sum of the squares of the distance of the vectors with respect to the mean. In other words,

$$\vec{r}_G = \operatorname{argmin}_{\vec{x}} \sum_i d^2(\vec{x}, \vec{r}_i). \quad (21)$$

Written in this form, we can solve for the centroid of a cluster using a general distance metric and arbitrary parameterization:

$$c_k = \operatorname{argmin}_x \sum_i d^2(x, p_i), \quad (22)$$

where for cluster k , c_k is the centroid and p_i are elements of the cluster.

Scaled \vec{h} - \vec{e} Vectors

Given the complexities involved with using the Q-law in clustering, the motivation exists to develop a metric that satisfies all metric properties and also uses a vector space to describe the orbits.

The angular momentum vector, \vec{h} , and eccentricity vector, \vec{e} , together completely describe the size, shape and orientation of an orbit (phase-free). It is equivalent to using the set of orbital elements semi-major axis, eccentricity, inclination, argument of periapsis, and right ascension of ascending node (a , e , i , ω , and Ω), but without concerns about angular singularities.

One could imagine performing clustering using the 6-vector obtained by concatenating the angular momentum vector and eccentricity vector together. This approach doesn't work well because of the drastic difference in magnitude of eccentricity ($0 \leq e < 1$) and the angular momentum. Also, it doesn't take into account how easy or difficult it is to change a given orbit. For example, it take less energy to change the angular momentum a certain amount for a larger orbit than a smaller one.

We propose a scaled version of the $[\vec{h}, \vec{e}]$ vector, where the scale factors are determined by how easily the vectors are changed when an external acceleration is applied. Let the applied acceleration be \vec{f} , the position be \vec{r} , velocity be \vec{v} , and the standard gravitational parameter be μ . The equations of motions are then:

$$\ddot{\vec{r}} = -\frac{\mu}{r^3}\vec{r} + \vec{f}. \quad (23)$$

The angular momentum and eccentricity vectors are:

$$\vec{h} = \vec{r} \times \vec{v}, \quad (24)$$

$$\vec{e} = \frac{\vec{v} \times \vec{h}}{\mu} - \frac{\vec{r}}{r}. \quad (25)$$

Under the applied acceleration, the derivatives of \vec{h} and \vec{e} are:

$$\dot{\vec{h}} = \vec{r} \times \dot{\vec{v}} = S(\vec{r})\vec{f}, \quad (26)$$

$$\dot{\vec{e}} = \frac{1}{\mu} \left(\dot{\vec{v}} \times \vec{h} + \vec{v} \times (\vec{r} \times \dot{\vec{v}}) \right) = \frac{1}{\mu} \left(-S(\vec{h}) + S(\vec{v})S(\vec{r}) \right) \vec{f}. \quad (27)$$

Equations (26) and (27) serve the same purpose as the Gauss equations. The matrix $S(\vec{z})$ is the skew-symmetric cross-product operator:

$$S(\vec{z}) = \begin{bmatrix} 0 & -z_3 & z_2 \\ z_3 & 0 & -z_1 \\ -z_2 & z_1 & 0 \end{bmatrix}.$$

Next, we find the maximum amount of change in \vec{h} that could be achieved by a given applied acceleration:

$$f_h = \max_{\vec{f}} \frac{\|\dot{\vec{h}}\|}{\|\vec{f}\|} = \max_{\vec{f}} \frac{\|S(\vec{r})\vec{f}\|}{\|\vec{f}\|} = \max_{\vec{f}.s.t.\|\vec{f}\|=1} \|S(\vec{r})\vec{f}\| \equiv \|S(\vec{r})\| = \|\vec{r}\| = r, \quad (28)$$

and for \vec{e} :

$$f_e = \max_{\vec{f}} \frac{\|\dot{\vec{e}}\|}{\|\vec{f}\|} = \frac{1}{\mu} \|-S(\vec{h}) + S(\vec{v})S(\vec{r})\| = \frac{1}{\sqrt{2}\mu} \sqrt{r^2v^2 + 4h^2 + rv\sqrt{r^2v^2 + 8h^2}}. \quad (29)$$

Equations (28) and (29) give the sensitivity of \vec{h} and \vec{e} to \vec{f} at any instance along the orbit. Next, we averaged these sensitivities over the orbit using true anomaly as the independent variable. One could also choose the maximum value over an orbit, such as in the Q-law. Equation (28) is straightforward to average:

$$\overline{f_h} = \frac{1}{2\pi} \int_0^{2\pi} f_h d\theta = a, \quad (30)$$

where θ is the true anomaly and a is the semi-major axis. The averaging of Eq. (29) is not as straightforward, and we used numerical quadrature to evaluate

$$\overline{f_e} = \frac{1}{2\pi} \int_0^{2\pi} f_e d\theta. \quad (31)$$

Now to the actual scaling. For each orbit, we are given \vec{h} and \vec{e} , then we compute $\overline{f_h}$ and $\overline{f_e}$. Next, we create the scaled vectors:

$$\begin{aligned} \tilde{h} &= \frac{\vec{h}}{(\overline{f_h})} \\ \tilde{e} &= \frac{\vec{e}}{(\overline{f_e})} \end{aligned}$$

Finally, the distance between two orbits, i and j , is simply the Cartesian distance between $[\tilde{h}_i, \tilde{e}_i]$ and $[\tilde{h}_j, \tilde{e}_j]$. By using a well-known distance metric, we ensure that the metric properties in Table 1 are satisfied. For clustering, the 6-element vectors $[\tilde{h}, \tilde{e}]$ are fed directly to the algorithms.

For a concrete example, Table 2 shows the angular momentum and eccentricity vectors for the first five points in the GTOC-5 contest. Table 3 shows the scaled values for the same orbits. Notice that the units of the scaled vectors are velocity – km/s in this case. This means that the distance between two scaled vectors can be interpreted as an approximation to the ΔV required to transfer between the two. Also, the scaling ensures that \tilde{h} and \tilde{e} are relatively close in magnitude, which is beneficial to clustering algorithms.

RESULTS

All 7,075 orbits from the GTOC-5 set are shown in three different ways in Figure 8. First, in the top-left is the a, e, i space, in the top-right are the angular momentum vectors, and in the bottom-left are the eccentricity vectors. Each orbit corresponds to one blue dot. Interspersed with the blue dots are black dots, which are elements of one of the 100 clusters produced by k -means using the scaled

Table 2. Example raw h - e vectors

Asteroid ID	h_x (km ² /s)	h_y (km ² /s)	h_z (km ² /s)	e_x	e_y	e_z
1	-4.704e+08	-5.917e+08	6.873e+09	0.3128	-0.0416	0.0178
2	-8.106e+08	-5.544e+08	5.134e+09	-0.1219	0.1866	0.0009
3	-8.512e+07	1.199e+09	5.876e+09	0.5187	-0.1857	0.0454
4	8.764e+08	3.287e+08	5.682e+09	-0.1081	0.5563	-0.0155
5	-1.601e+09	2.239e+09	5.475e+09	0.4982	-0.0759	0.1767

Table 3. Example scaled h - e vectors

Asteroid ID	\tilde{h}_x (km/s)	\tilde{h}_y (km/s)	\tilde{h}_z (km/s)	\tilde{e}_x (km/s)	\tilde{e}_y (km/s)	\tilde{e}_z (km/s)
1	-1.1754	-1.4785	17.1733	2.9755	-0.3956	0.1696
2	-3.7415	-2.5590	23.6970	-1.5401	2.3580	0.0115
3	-0.2180	3.0714	15.0523	5.5628	-1.9916	0.4870
4	2.3795	0.8925	15.4290	-1.2053	6.2013	-0.1728
5	-4.0431	5.6568	13.8294	5.2432	-0.7991	1.8597

\vec{h} and \vec{e} vectors. This particular cluster contains 67 orbits, which are shown in the bottom-right. The black dots on this subplot mark the periapsis location of each orbit, and the thick black orbit is the centroid of the cluster.

CONCLUSIONS AND FUTURE WORK

We presented four clustering algorithms for the automatic classification of celestial body orbits in terms of their navigation proximity: k -means, fuzzy c -means, expectation maximization, and graph partitioning.

To the extent of our knowledge, despite the widespread adoption of these algorithms in many analytical sciences, this work represents the first attempt at applying large-scale data clustering techniques for target identification in the context of spacecraft trajectory optimization.

The central challenge for any clustering methodology lies in the definition of *proximity* between two orbits, and in the adequacy of using such proximity measure as a metric space in the strict sense. Another difficulty in using any of the clustering methods discussed in this paper is the need to provide the number of clusters *a priori*. These algorithms are sensitive to such selection, and should be run several times with different numbers of clusters to characterize the underlying data set.

In addressing the shortcomings of k -means, we introduced other methodologies that ease the *hard* partitioning characteristic of the original algorithm. Two are based on statistical assignment: fuzzy c -means, which measures the membership of item i in cluster j as a probability in terms of a *weight* w_{ij} , and expectation maximization—an extension of fuzzy c -means—where the weight w_{ij} is updated iteratively until convergence to a Normal Distribution. We also introduced a methodology that does not depend directly on a measure of proximity. Instead, the graph-partitioning algorithm relies on representing the assignment problem as a graph, where the nodes are the orbits, and edges are added iteratively based on graph algorithms like Delaunay triangulation, convex hull, or multi-level partitioning.

We also provided two measures of proximity which provide better results than a naïve direct comparison among orbital elements: the proximity quotient and modifications thereof from the

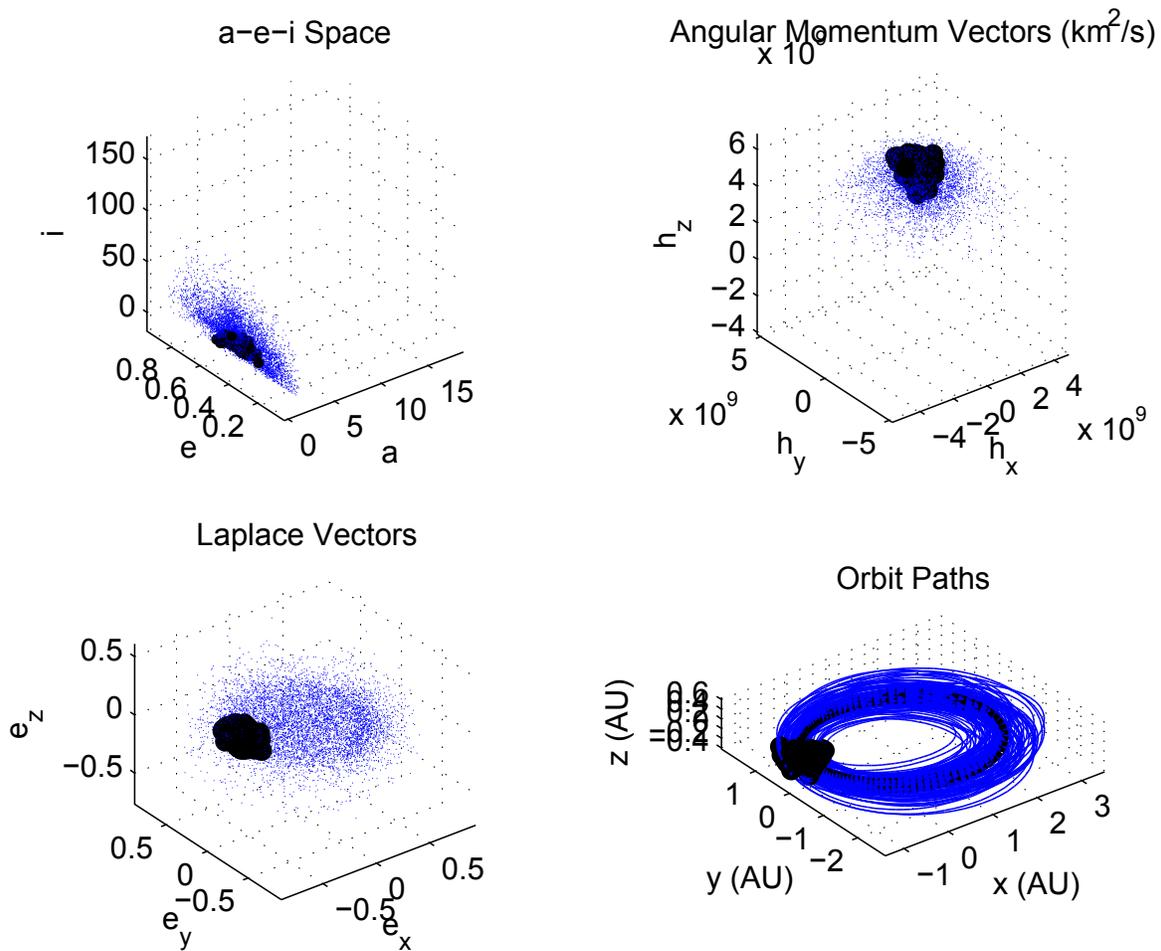


Figure 8. Example of k -means clustering using dynamically-scaled angular momentum and eccentricity vectors

Q-law, and a metric in the $\vec{h} - \vec{e}$ space.

We are encouraged by the results obtained so far, and have already benefited from the application of these algorithms and metrics in the challenging GTOC competition. Our future lines of research include: (1) providing a technique for the automatic classification of orbital debris, (2) introducing a generalized metric space for sets of orbits, and (3) consolidating the strengths of the algorithms presented into one general purpose classification mechanism.

ACKNOWLEDGEMENTS

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

© 2013 California Institute of Technology. Government sponsorship acknowledged.

REFERENCES

- [1] A. E. Petropoulos, “Refinements to the Q-law for the Low-Thrust Orbit Transfers,” *15th AAS/AIAA Space Flight Mechanics Conference*, January 2005.
- [2] J. MacQueen, “Some Methods for Classification and Analysis of Multivariate Observations,” Proc. 5th Berkeley Symp. Math. Stat. Probab., Univ. Calif. 1965/66, 1, 281-297 (1967), 1967.
- [3] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. Norwell, MA, USA: Kluwer Academic Publishers, 1981.
- [4] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum Likelihood from Incomplete Data via the EM Algorithm,” *Journal of the Royal Statistical Society. Series B (Methodological)*, Vol. 39, No. 1, 1977, pp. 1–38.
- [5] D. Arthur and S. Vassilvitskii, “k-means++: The Advantages of Careful Seeding,” *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '07*, Philadelphia, PA, USA, Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.
- [6] M. R. Gupta and Y. Chen, “Theory and Use of the EM Algorithm,” *Found. Trends Signal Process.*, Vol. 4, Mar. 2011, pp. 223–296, 10.1561/20000000034.
- [7] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, “The Quickhull Algorithm for Convex Hulls,” *ACM Trans. Math. Softw.*, Vol. 22, Dec. 1996, pp. 469–483, 10.1145/235815.235821.
- [8] G. Karypis and V. Kumar, “A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs,” *SIAM J. Sci. Comput.*, Vol. 20, Dec. 1998, pp. 359–392, 10.1137/S1064827595287997.