CLUSTERING OF LARGE EVENT SETS USING DELAUNAY TESSELLATIONS AND K-MEDOID OPTIMIZATION

James R. Hipp and Christopher J. Young, Sandia National Laboratories

Sponsored by National Nuclear Security Administration, Office of Nonproliferation Research and Engineering, Office of Defense Nuclear Nonproliferation, Contract No. DE-AC04-94AL85000

SAND2007-6027C

Introduction

Many technical fields such as spatial data mining and operations research are interested in the problem of clustering arbitrary unstructured geo-referenced data sets. Most of these spatial clustering techniques make use of data proximity. In these methods proximity is based on a selected Euclidean metric which attempts to capture the spatial autocorrelation and near association of neighboring spatial data. Two general approaches are typically followed to form the clusters: *a)* a bottom-up approach which agglomerates spatial points to form clusters based on their relative nearness to one another; and *b)* a top-down approach which attempts to partition a heterogeneous data set into smaller more similar groups.

Of particular interest for clustering seismic events are top-down medoid-based clustering methods. In these methods the point representing an arbitrary cluster must be chosen from the set of events forming the cluster. Other methods include those based on discovering a mean, or average, location within the cluster to use as a cluster representative. Mean-based methods are generally more popular and possess fast solution algorithms of order *O(n)* time. However, the calculated mean representative locations are not necessarily near the cluster center when outliers are involved in the calculation and suffer many other statistical bias and consistency problems. A good description of the advantages and disadvantages of mean- and medoid-based methods can be found in Estivill-Castro et al. (2001).

The medoid (or K-Medoid where K refers to the number of clusters) approach suffers two primary drawbacks. The first, which it shares with mean-based methods, is that the number of clusters must be defined in advance. The second is that for large data sets containing up to *n* points the solution times are of order $O(n^2)$.

In this paper we shall show a method that reliably calculates the number of required clusters and that utilizes spatial based proximity information embedded within the Delaunay tessellation (Delaunay, 1934) of the event set to improve performance to *O(n log n)*. Since code to construct a spherical Delaunay tessellator had already been developed for the GNEMRE program, only the K-Medoid clustering algorithm needed to be developed.

Algorithm Overview

Assume an initial distribution of geo-referenced events located in an unstructured fashion anywhere on the surface (or near surface) of the Earth. Our goal shall be to find clustered sets of those events that share close spatial proximity and are no more distant from one another (in a cluster) than some user prescribed distance $\boldsymbol{D_{max}}$. We shall call $\boldsymbol{D_{max}}$ the desired cluster size parameter. The set of discovered clusters shall be returned as an array of cluster objects each containing a list of one or more events, which defines the cluster's event set, and a single event, the cluster representative, which signifies the best characterization of the cluster.

Before proceeding further we shall define the concepts of an *event group* (i.e. cluster), the *maximum span* of a cluster, and a *consistent* cluster which will be used repeatedly in the discussion that follows:

- A group of events (or *event group*) shall be defined such that for any event contained in the group there is a closest neighbor event, also contained in the group that is never further away than some distance $\boldsymbol{D_{max}}$. Or conversely, two event groups are distinct if all events in the first group are further away than $\boldsymbol{D_{max}}$ from all events in the second group.
- The *maximum span* of a cluster (event group) is the furthest distance between any two points in the cluster, which by definition is less than or equal to $\boldsymbol{D_{max}}$.
- A cluster is said to be *consistent* if all events contained by an arbitrary cluster are closer to the cluster's representative event than to any other representative in the remaining set of adjacent clusters of the event group.

The clustering approach defined in this paper begins by identifying isolated event groups from the initial set of globally distributed events. Next, each event group is processed sequentially by sub-dividing the group into an initial "best" guess of consistent clusters where the number of clusters is chosen in a minimalist way while still guaranteeing that the maximum span of each cluster is not violated. Finally, the group is processed using a top-down k-medoid algorithm to find the best set of cluster representatives for the group.

The remaining pages of this paper shall describe the process in reverse order. First we will describe the k-medoid optimization assuming an event group of an initial set of clusters has been found and representatives have been assigned. Next we will return to describe the method for forming the initial best guess of clusters and their representatives given an arbitrary overall event group. Finally, we will describe the process for forming event groups given the initial distribution of globally geo-referenced events. The actual process includes one additional step after k-medoid optimization to handle the rare small fix-ups needed where poor quality clusters are formed and events are assigned to inappropriate clusters, but we will not describe that here.

K-Medoid Optimization

The K-Medoid clustering optimization algorithm attempts to improve an initial clustering definition for an event group that has been decomposed into a consistent set of initial "best" guess clusters. Figure 1 illustrates an example event group composed of 4 initial clusters whose initial representatives are outlined in red. The sequence of optimization steps for this example are shown in Figure 2a-d. The optimization heuristic operates by iteratively discovering a new set of representatives for the clusters that globally minimizes

$$M(C) = \sum_{j=0}^{N_c-1} \sum_{i=0}^{N_j-1} w_i d_{ki}$$

where the sum is taken over all events (N_j) in the event group. The value d_{ki} represents the distance between the i^{th} event and the k^{th} cluster representative. The weights can be used to influence the minimization by defining a criterion that represents something other than proximity. For purposes of the remainder of this paper the weights will be assumed to be one for all events such that *M(C)* is strictly proximity based.

We intend to find the global minimum subject to two constraints:

a) no event-to-event distance span in the cluster shall exceed the pre-defined maximum distance, $\boldsymbol{D_{max}}$; and

b) no two clusters from the group can be combined to form a new cluster whose maximum span is also < $\boldsymbol{D_{max}}$.

The first of these constraints is simply the definition, previously given, for the maximum allowable span of any cluster. The second definition is a similar attempt to control the minimum size of a cluster so that is not significantly smaller than $\boldsymbol{D_{max}}$. The first constraint is used during the optimization to prohibit new representatives from being formed that will cause an excessive span even if the value of *M(C)* is smaller given the new representative for some cluster. The second constraint ensures that adjacent clusters that can be represented by a single cluster, assuming the first constraint is not violated, are merged into a single cluster reducing the total number of clusters in the group by one.

With the aforementioned constraints and definitions we define the K-Medoid optimization as follows:

Given an initial set of representative clusters, fixed in number, from an arbitrary event group, exchange non-representative events with representatives until a reduced value for M(C) is discovered. When a more optimum event is discovered swap the old representative with the new event and continue with the next event in the group repeating the test. Continue until none of the non-representative events reduces the value of M(C) more than the current set of representatives subject to the previously defined constraints.

Figure 2a shows an arbitrary iteration of the algorithm where event n_i is to be tested to see if it is a better representative for cluster *j* than its current representative. The change in *M(C)* caused by replacing the current representative of cluster *j* with event n_i is composed of two components. These components include:

a) Those events that currently belong to clusters other than *j* but are transferred to cluster *j'* when event n_i becomes its representative (let *j'* be the *jth* cluster when event n_i is made its representative); and

b) The original events of cluster *j* which may remain in cluster *j'* or may be transferred to other clusters when event n_i becomes the new representative.

Let's examine events of type a) first. For our example only one event, n_a, is transferred from a non-*j* cluster to cluster *j'* when n_i becomes the representative of cluster *j'*. The net change in *M(C)* as a result of the transfer of n_a into cluster *j'* is composed of the additional contribution of assigning n_a into cluster *j'* minus the removal of the contribution from the cluster that currently contains n_a. We'll let *r(a)* be the index of the cluster containing n_a before the swap.

This net change in *M(C)* is shown in Figure 2b. Because event n_a lies closer to n_i than its old representative (that is why it is being transferred) the net result is always negative. For this reason, events of type a) that occur in a k-medoid swap tests always produce a reduction in *M(C)*.

For events of type b) we need to examine the contributions to the change in *M(C)* from all of the events that reside in the current cluster *j*. In our example problem shown in Figure 2c, four of the original cluster *j* events will remain in cluster *j'*, while two of the original events will be moved to nearby adjacent clusters. The net change in the value of *M(C)* is caused by the loss of all contributions of cluster *j* events with the cluster *j* representative (red arrows) and a gain due to the reattachment of cluster *j* events with the cluster *j'* representative or other representatives other than *j'* (blue arrows). We can write these contributions over all cluster *j* events as

$$\sum_{\beta} w_\beta \left(d'_{m(\beta)\beta} - d_{j\beta} \right) = \sum_{\beta} w_\beta \left(\left| n_\beta - n_{C_{r(\beta)}} \right| - \left| n_\beta - n_{C_j} \right| \right)$$

where *b* is taken over all original events of cluster *j*. The new reassignment cluster indices are contained in *m(b)* for each event in cluster *j*.

The total change in *M(C)* is given by summing changes from both case a) and case b) which gives

$$\delta M(C) = \sum_{\beta} w_\beta \left(d'_{m(\beta)\beta} - d_{j\beta} \right) + \sum_{\alpha} w_\alpha \left(d'_{j\alpha} - d_{r(\alpha)\alpha} \right)$$

If the change in *dM(C)* is < 0.0 then event n_i represents a more optimum cluster representative than does the original cluster representative C_j and a swap is performed. This case is illustrated in Figure 2d. If, however, *dM(C)* is > 0.0 then representative C_j is better and the swap is not performed. In either case the algorithm advances to the next event n_i for further testing. The algorithm halts when no further swapping occurs after a complete pass through all events in the group.

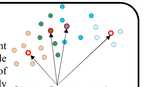

Figure 1. Event groups initial cluster definition.

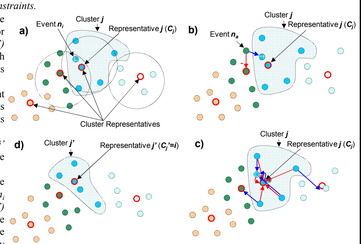

Figure 2. An example of the K-medoid optimization process.

K-Medoid Constraint Satisfaction

As previously discussed the k-medoid optimization is constrained to avoid clusters with spans that exceed D_{max} or spans that are unnecessarily small (much less than $D_{max}/2$). In the first case we enforce the constraint by ensuring that representative swaps are not performed during optimization if the swap results in a cluster whose span exceeds D_{max}. Figure 3 illustrates a case of two clusters before testing an event as a replacement representative for the topmost cluster. Notice that both clusters have a span that is less than D_{max} before the test.

Following the test the event is found to reduce *M(C)* (assume for the sake of the example) which results in the migration of an event from the cluster containing the test event to the lowermost cluster in order to ensure cluster consistency requirements (events must reside in the cluster for which they are closest to the cluster's representative). However, as shown in Figure 4 the event migration results in a span that exceeds D_{max}. For this case the swap is disallowed and the original configuration before the test is restored.

The second constraint is satisfied by attempting to merge adjacent clusters with one-another during the optimization process. If two clusters can be merged, as shown in Figure 5, where the resulting merger produces a new cluster whose span is still less than D_{max} then the merger is processed. The net result reduces the cluster count by one which generally increases the intra-cluster spacing.

Fig 3. Adj. cluster pairs before test representative swap.

Fig 4. After representative swap resulting in an excessive span.

Figure 5. Adjacent cluster merge reducing the cluster count by one.

Event Group Initial Cluster Definition

In this section we shall go back to the beginning to answer the question of how to form an initial set of clusters from a given event group. It is this initial cluster definition upon which the k-medoid algorithm operates.

For our specific problem we are interested in determining the number of clusters necessary to represent the events in a group given some desired size, D_{max} for each cluster. Here we shall define a method of approximating the cluster count and initial distribution using a medial axis sub-division scheme. As before, let the span of a cluster be the largest distance between any two events in the cluster. If the span of a cluster exceeds D_{max} then sub-divide the cluster into two new sub-clusters. In turn, evaluate each of the new sub-cluster spans. If one or both still exceed D_{max} then one or both are also split into two new sub-clusters. This process is repeated recursively until the newly formed sub-cluster pairs are defined by a set of events whose span is less than or equal to D_{max}.

The best way to sub-divide the events into sub-cluster pairs is to simply split the clusters event set along a line (or plane in 3-space) that is orthogonal to the line (or plane) that defines the clusters span and positioned so that it contains the span's mid-point. This new splitting axis is called the medial axis of the span. The sub-division process is illustrated in Figures 6a through 6c.

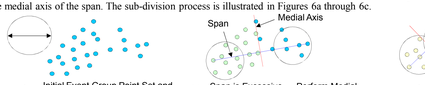

Figure 6. Medial-axis sub-division of an initial event group into 4 sub-clusters all of whose spans are < $\boldsymbol{D_{max}}$.

The span axes are denoted in blue while the medial axes are shown in red. Notice that the medial axis effectively sub-divides the cluster into two new clusters that lie on opposite sides of the medial axis. A simple test (scalar triple product in 3-space) can be used to determine which side of the medial axis an event lies. Events lying on either side of the medial axis are inserted into two new sub-clusters and the original cluster is removed. In the example above the original set of events in the group are subdivided into 4 sub-clusters whose spans are all less than D_{max}.

The previously described decomposition of the original event group into a set of clusters whose spans are all less than D_{max} determines the initial number of clusters that will be provided to the k-medoid clustering algorithm. However, it has not defined the best characteristic representative for each cluster which must be accomplished before performing the k-medoid optimization. This is done by simply performing a local optimization test for *M(C)* on each event in each cluster. The one that minimizes *M(C)* for each cluster is chosen as its representative. Figure 7a shows the result given the final clusters shown in Figure 6 above.

The final step before entering the k-medoid optimization algorithm is to ensure that the initial clusters are all consistent. This means that we must check each cluster to ensure that all events owned by the cluster are closest to that cluster's representative than to any other representative in the remaining clusters. For our example three events are found to be erroneously assigned within other clusters and must be reassigned. These reassigned events are shown in Figure 7b.

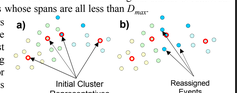

Figure 7. a) Event group initial cluster representative formations. b) Event reassignment required to preserve cluster consistency.

Event Group Discovery

We can now answer the final question of how to efficiently construct event groups given the raw event data distributed in an unstructured manner over the entire Earth's surface. Here we shall make direct use of the nearest neighbor information contained in a Delaunay tessellation of the raw event positions.

Since we already have code to construct a spherical Delaunay tessellator we need only input the event positions and build the tessellation. Figure 8 shows the resultant edge connectivity between ISC catalog events for the year 2000 with depth <= 33 km and m_b >=3.0. The input set contains 22,766 events. The small white square near Greece will be shown in the remaining figures to illustrate the process steps.

Event groups are defined by finding connected sets of events where the distance between neighboring events in the connected set is less than D_{max}. This can be accomplished in order *O(n)* time using the topology of the Delaunay tessellation. The algorithm simply picks the first unmarked node (event) in the tessellation. By marking we are simply setting a flag to indicate that the node has been visited. If the node is already marked we proceed to the next node. This continues until no unmarked nodes remain.

Any unmarked nodes are processed by first marking the node as having been visited and adding it into a new event group. We then add the node to a stack. Next the algorithm enters a loop processing all nodes in the stack until it is empty. The stack processing begins by first popping the next node off of the stack which is taken as the current process node. Then we loop over each edge of the node (the tessellation topology stores this information) and check the edge length to see if it is less than or equal to D_{max}. If it is and the node opposite the current process node on the edge has not been flagged as visited we mark it as such and add that opposite node to the stack and to the current event group (a cluster). This continues for each edge of the current process node. Finally the algorithm checks to see if any nodes remain in the stack and loops to obtain the next current process node if the stack is not empty.

When the stack is empty the algorithm returns to the outer loop over all tessellation nodes to find the next unmarked node. When no unmarked nodes remain the event groups have all been discovered. Each group consists of one or more events. We can best visualize these by turning off all edges that exceed D_{max}. This is shown in Figure 9 for a D_{max} value of ½ degree. The groups are highlighted with an outlined yellow curve to aid in their visualization.

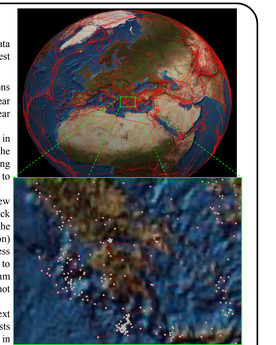

Fig 8. 22,766 spherically tessellated ISC events.

Event groups whose total span is less than D_{max} form completed clusters. Groups whose span exceeds D_{max} must be processed by the medial-axis sub-division algorithm to form initial consistent clusters which are further processed by the k-medoid clustering algorithm to find the best set of representatives in the group.

In Figure 10 all isolated events and groups of 2 events are immediately considered as completed. The remaining groups must be processed by the medial-axis subdivision and k-medoid algorithms. The result of that processing is shown in Figure 11. The cluster representatives are outlined in white while the clusters are signified by the red circles centered on the maximum span of each cluster.

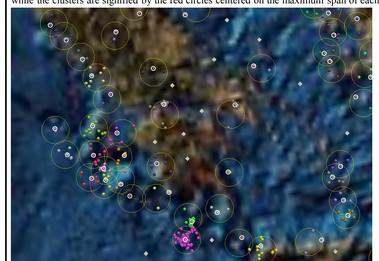

Fig 10. Magnified region depicting final cluster formation following group formation, initial cluster definition, and k-medoid optimization. White circles indicate cluster representatives. Red circles are D_{max} in diameter centered on cluster max spans. For clarity, single event groups are shown as white diamonds.

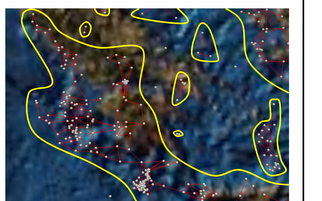

Fig 9. Local event groups and isolated events in the zoom region.

The final total clustering algorithm can be summarized as follows:

1) Tessellate raw events globally on a sphere
2) Decompose events into groups whose neighbor separation is < D_{max}
3) Move groups whose span is < D_{max} into a completed cluster list (includes isolated events)
4) For each remaining incomplete event group
   {
   >>Decompose group into a set of sub-clusters using the medial-axis sub-division algorithm
   >>Perform k-medoid representative optimization on entire group
   >>Fix poor quality clusters and event assignments
   >>Add each optimized sub-cluster to the completed cluster list
   }

REFERENCES

Estivill-Castro, V, Houle, M. E., (2001), Robust Distances-Based Clustering with Applications to Spatial Data Mining, Algorithmica, 30(2):216-242.

Delaunay, B.N., (1934), Sur la Sphere Vide, Bull. Acad. Science USSR VII: Class. Sci. Math., 793-800.

Sandia is a multi-program laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.