Multi-Layer Increasing Coherence Clustering of Autonomous Systems with MULICsoft

Bill Andreopoulos

Aijun An

Xiaogang Wang

Technical Report CS-2005-07

April 2005

Department of Computer Science and Engineering

4700 Keele Street North York, Ontario M3J 1P3 Canada

Multi-Layer Increasing Coherence Clustering of Autonomous Systems with MULICsoft

Bill Andreopoulos
Department of Computer Science, York University, Toronto, Ontario, Canada, M3J 1P3
billa@cs.yorku.ca

Aijun An
Department of Computer Science, York University, Toronto, Ontario, Canada, M3J 1P3
aan@cs.yorku.ca

Xiaogang Wang
Department of Mathematics and Statistics, York University, Toronto, Ontario, Canada,M3J1P3
stevenw@mathstat.yorku.ca

ABSTRACT

Discovering the properties of the Internet topology is crucial for the use, maintenance and optimization of the Internet. A recent approach in the research community is to collect IP path data from active probing of hosts and to map the IP paths to known Autonomous System (AS) paths. We apply the MULICsoft clustering algorithm to network data for ASs forming the Internet backbones. MULICsoft is intended for categorical data sets where each categorical attribute value (CA) has a 'weight' in the range 0.0 to 1.0, indicating how strongly the corresponding CA should influence the clustering process. MULICsoft produces as many clusters as it can find in the data set. Each cluster consists of layers formed gradually through iterations, by reducing the similarity criterion for inserting objects in layers of a cluster at different iterations. The objects in the data are ASs and the CAs represent links between ASs. The weights are inversely related to the number of unknown IPs in an indirect link between ASs. We introduce a novel significance metric for identifying the most significant ASs in a cluster.
http://www.cs.yorku.ca/~billa/MULIC/

1. INTRODUCTION

Clustering aims to partition a set of objects into groups, so that objects with similar characteristics are grouped together and different groups contain objects with different characteristics. A high quality clustering tool produces clusters with *high intra-class* similarity and *low inter-class* similarity [10-13]. Clustering is applied to Internet data to help researchers identify prominent subgroups of the Internet map and understand the structure of the Internet. Clustering techniques employ certain types of Internet traffic data sets - such as packets sent between network points - to identify significant subgroups. A subgroup in an Internet data set generally represents a set of network points that communicate with similar sets of network points through sending and receiving packets [3-6, 8, 15, 16]. Our objective is to partition an Internet traffic data set into clusters, so that network points sending and receiving packets to and from similar sets of points are placed in the same cluster, while points in different clusters send and receive packets to and from dissimilar sets of points [3, 8, 15, 16].

We developed the MULICsoft categorical clustering tool, which is based on the MULIC algorithm that is described in [2]. We showed that MULIC clustering results are of higher quality than those of other categorical clustering algorithms, such as k-Modes, ROCK, AutoClass, CLOPE and others [2]. Characteristics of MULIC and MULICsoft include: **a.** The algorithm does not sacrifice the coherence of the resulting clusters for the number of clusters desired. Instead, it produces as many clusters as there naturally exist in the data set. **b.** Each cluster consists of layers formed gradually through iterations, by reducing the similarity criterion for inserting objects in layers of a cluster at different iterations.

Section 2 describes the AS network data set. Section 3 gives an overview of previous related work in network data clustering. Section 4 outlines the MULICsoft categorical clustering algorithm and the extensions that we have built for clustering network data. Section 5 describes the results. Section 6 proposes a novel significance metric for the clustering results. Section 7 concludes the paper.

2. DESCRIPTION OF DATA SETS

An AS is a collection of IP networks under control of a single entity, typically an Internet service provider (ISP) or a very large organization with redundant connections to the rest of the Internet [5]. Because of the large size of the Internet, researchers often restrict analysis of the Internet to the AS level. We have applied the MULICsoft clustering algorithm to the AS data set provided by the Cooperative Association for Internet Data Analysis (CAIDA) [3-5, 15, 16]. This data set is derived from IP addresses collected using a TCP utility called traceroute that are mapped to ASs. The data reflects packets that have traversed a path, representing links between ASs. We chose this data as it is more likely to faithfully correspond to IP topology than Border Gateway Protocol (BGP) data [5].

Each object in the data set corresponds to an AS. The CAs represent the relationships or links between ASs (packets passed from an AS to other ASs). The data set contains a large number of 'zeros' with 'ones' occurring sparsely. There are 12,517 objects in the data set, corresponding to 12,517 ASs on the Internet [5]. The AS data set contains:

Direct links between ASs: A direct link from source AS to destination AS exists if an IP was found in the source AS's address space and it was directly followed by an IP in the destination AS's address space.

Indirect links between ASs: An indirect link from source AS to destination AS exists if an IP was found in the source AS's address space, followed by a number of IPs with an unknown prefix, ended by an IP in the destination AS's address space [5]. We use the number of unknown IP addresses between source and destination ASs, k, to compute the 'weights' on the categorical attribute values. The weights are computed by evaluating $1/(k+1)$. Thus, a k value of 0 becomes a weight of 1.0, a k value of 1 becomes a weight of 0.5, a k value of 3 becomes a weight of 0.25 and so on.

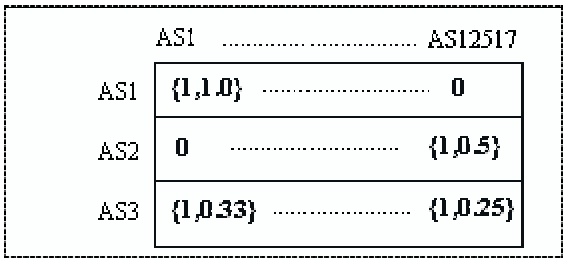

Fig. 1 - The AS data set to be clustered. Each cell representing a link has a value of {1, weight}.

We used categorical data for clustering by representing the links between ASs as a matrix shown in Figure 1, where a value of "one" in a cell represents a link between ASs and a value of "zero" in a cell represents no link. We imposed weights on the CAs of "one", representing the number of unknown IP addresses between ASs. We used a special similarity formula described in Section 4.5.

3. RELATED WORK

Graph theory was applied by Faloutsos et al. [9] to AS Internet data for discovering properties of the Internet topology, such as power-laws that govern AS graphs. Chang et al. [6] included additional data in the analysis to depart from the power-law discovery in AS graphs. Vukadinovic et al. [22] found that the Normalized Laplacian Spectrum (NLS) of AS graphs is invariant regardless of the Internet's exponential growth. Mihail et al. [18] used eigenvectors corresponding to the largest eigenvalues of the Laplacian matrix to find AS clusters with certain characteristics, such as geographic locations. Chen et al. [7] employed NLS and applied it to AS data from CAIDA and from the Route Views project to identify cluster characteristics.

Clustering methods have been applied to network data analysis in Intrusion Detection Systems (IDSs), when the log files of user behavior are too large for an expert to analyze [24]. Clustering is applied to files with logged behavior of users over time, to separate instances of normal activity from instances of abusive or attack activity. Both unsupervised and supervised clustering can be used.

Unsupervised clustering: When unsupervised clustering is applied to IDSs' data on which no prior knowledge exists, the data is first clustered and then different clusters are labeled as either 'normal' or 'intrusions' based on the size of the cluster. Then, a new unclassified object is classified in the cluster to which it is the closest. Given a new object d, classification proceeds by finding a cluster C which is closest to d and classifying d according to the label of C as either normal or anomalous [17, 19].

IDSs based on unsupervised clustering focus on activity data that may contain features of an individual TCP connection, such as its duration, protocol type, number of bytes transferred and a flag indicating the connection's normal or error status. Other features may include the number of file creation operations, number of failed login attempts, whether root shell was obtained, the number of connections to the same host as the current connection within the past two seconds, the number of connections to the same service as the current connection within the past two seconds and others.

Clustering of the data sets is done using a typical single-linkage clustering. This is not the most effective type of clustering but it is very fast with a complexity of O(n).

Then, clusters are labeled based on the size of the cluster, as clusters containing normal instances or clusters containing attacks. Since normal instances constitute a majority of the data set, clusters with more objects are more likely to contain normal instances than attacks. Therefore, the clusters containing the largest number of objects are labeled "normal" while the rest of the clusters are labeled "anomalous". A problem with this approach is that many sub-types of normal objects may exist in the data set – such as using different protocols like ftp, telnet, www, etc – in which case a large number of clusters might be produced containing a small number of normal objects, one for each type of normal use of the network. Then, these normal clusters may be incorrectly labeled as "anomalous".

Solka et al. [20] examine the application of hierarchical cluster analysis to the characterization of single machine SYN ACK time series. The purpose of this analysis is the identification of normal and abnormal activity for a small group of mail and web servers.

Supervised classification: Supervised classification is applied to IDSs for Signature Recognition purposes. This means that signatures representing patterns of previous intrusion activities are learned by the system. Then, new patterns of activity are compared to the previously learned signatures to determine if the activities are normal or if they may be intrusions [23].

The automatic learning of intrusion signatures from historic data containing labeled examples of normal and intrusive activity can be done using a data mining technique such as Support Vector Machines. The learned intrusion signatures should be updated whenever more data of normal and intrusive activity becomes available.

IDS based on signature recognition focus on two main types of activity data: network traffic data and computer audit data. Some categorical activity attributes that can be obtained from this data include the user id, event type, process id, command, remote IP address. Some numerical attributes that can be obtained from this data include the time stamp, CPU time etc.

Using SVMs, new activity patterns are matched to the learned signatures to determine if they are more likely to fall in the class of normal or intrusive patterns. An example is the CCA-S clustering algorithm used for IDSs [23].

4. MULICsoft CLUSTERING

Clustering is performed using MULICsoft – an extension of MULIC [2] – having a special similarity metric that is described in Sections 4.4 and 4.5. Figures 2 and 3 illustrate the algorithm and the results, respectively. MULICsoft is an extension of the k-Modes clustering algorithm for categorical data sets [13]. k-Modes is a clustering algorithm that deals with categorical data [13]. The k-Modes clustering algorithm requires the user to specify from the beginning the number of clusters to be produced and the algorithm builds and refines the specified number of clusters. Each cluster has a mode associated with it. Assuming that the objects in the data set are described by m categorical attributes, the mode of a cluster is a vector $Q=\{q_1, q_2, \ldots, q_m\}$ where q_i is the most frequent value for the ith attribute in the cluster of objects.

MULICsoft makes substantial changes to the k-Modes algorithm. The purpose of our clustering algorithm is to maximize the following similarity formula *at each iteration individually*, while at the same time to ensure that all objects can eventually be inserted in clusters if so desired:

$$\sum_{i=1}^{N} similarity(o_i, \text{mode}_i) \qquad (1)$$

where o_i is the ith object in the data set and mode_i is the mode of the ith object's cluster containing the most frequent values in the cluster. Maximizing formula (1) leads to the situation where all objects are as similar to their clusters' modes as possible, thus minimizing the number of values in a cluster that differ from the most frequent value for the same attribute.

The MULICsoft algorithm has a few more requirements. First, the number of clusters is not specified by the user. Clusters should be formed, removed or merged during the clustering process as the need arises. Second, a cluster *c* must contain at least two objects, otherwise, it is not a cluster. Third, it must be possible for all objects to be inserted in clusters by the end of the clustering process if so desired.

MULICsoft normally is preceded by a preprocessing step, where the data objects are ordered according to the frequency by which their CAs appear in the data set – details of this step are given in [2]. The main part of the MULICsoft clustering algorithm is shown in Figure 2. The algorithm starts by reading all objects from the input file and putting them in order. Then, it continues iterating over all objects that have not been placed in clusters yet, to find the closest cluster. In all iterations, the closest cluster for each unclassified object is determined by comparing how many similar attributes exist between a mode and the object. The similarity between a mode and an object is determined by the similarity equation described in Section 4.5.

The variable *num_values_can_differ* is maintained to indicate how strong the similarity has to be between an object and the closest cluster's mode for the object to be inserted in the cluster – initially *num_values_can_differ* equals 0, meaning that the similarity has to be very strong between an object and the closest cluster's mode. If the number of different values between the object and the closest cluster's mode are greater than *num_values_can_differ* then the object is inserted in a new cluster on its own. If the number of different values between the object and the closest cluster's mode are less than or equal to *num_values_can_differ*, then, the object is inserted in the closest cluster and the mode is updated.

At the end of each iteration, all objects classified in clusters with size one have their clusters removed so that they will be considered again at the next iteration. This ensures that the clusters that persist through the process are only those containing at least 2 objects for which a substantial similarity can be found. Objects belonging to clusters with size greater than one are removed from the set of unclassified objects.

At the end of each iteration, if no objects have been placed in clusters of size greater than 1, then the variable *num_values_can_differ* is incremented to represent how many values are allowed to differ next time. Thus, at the next iteration the threshold will be more flexible, ensuring that more objects will be placed in clusters. It is possible for all objects to be classified eventually, even if the closest cluster is a little similar. The iterative process stops when all objects have been placed in clusters of size greater than 1, or when *num_values_can_differ* exceeds a user-specified threshold.

The MULICsoft algorithm can eventually place all objects in clusters, because *num_values_can_differ* can continue increasing until all objects are classified. Even if, in the extreme case, an object *o* with *I* attributes has only one value similar to the mode of the closest cluster, it can still be classified when *num_values_can_differ* = *I-1*.

Figure 3 illustrates what the results of MULICsoft look like. Each cluster consists of many different "layers" of objects. The layer of an object represents how strong the object's similarity was to the mode of the cluster when the object was allocated to it. The cluster's layer in which an object is inserted depends on the value of *num_values_can_differ*. Lower layers have a lower coherence and correspond to higher values of *num_values_can_differ* and to a more flexible similarity criterion for insertion. MULICsoft starts by inserting as many objects as possible in high layers – such as layer 0 or 1 - and then moves to lower layers, creating them as the need arises. Eventually, all objects can be classified in clusters; if little similarity exists between an object and its closest cluster mode, the object can be inserted in a low layer of the cluster.

If an unclassified object has equal similarity to the modes of the two (or more) closest clusters, then the algorithm tries to resolve this 'tie' by comparing the object to the modes of the clusters' top layers. These modes were stored by MULICsoft when the clusters were created, so they do not need to be recomputed. If the object has equal similarity to all top layers' modes, the object is assigned to the cluster with the highest bottom layer. If all clusters have the same bottom layer then the object is assigned to the first cluster, since there is insufficient data for selecting the best cluster.

Input: (1) a set *S* of objects;
(2) the maximum number of values that can differ between an object and the mode of its cluster (*threshold* for *num_values_can_differ*)
Output: a set of clusters
Method:
Order the objects in *S* according to their scores [2];
Insert the highest-ordered object into a new cluster, use the object as the mode of the cluster, and remove the object from *S*;
Initialize *num_values_can_differ* to 0;
Loop through the following until S is empty or *num_values_can_differ* is greater than the specified *threshold*
For each object *o* in *S* from the highest-ordered to lowest-ordered
Find *o*'s closest cluster *c* by comparing *o* with the modes of all existing cluster(s);
If the number of different values between *o* and *c*'s mode is larger than *num_values_can_differ*, insert *o* into a new cluster
Otherwise, insert *o* into c and update *c*'s mode;
Remove object *o* from *S*;
For each cluster *c*, if there is only one object in *c*, remove *c* and put the object back in *S*;
If in this loop no objects were placed in cluster with size > 1, increment *num_values_can_differ* by 1.

Fig. 2 - Main part of the MULICsoft clustering algorithm

The runtime complexity of MULICsoft is *O(IN)*, where *I* represents the number of annotations in an object and *N* represents the number of objects in the data set [2].

4.1 Optional Merging of Clusters

We should generally avoid the situation where the similarity of the top layers of two different clusters is stronger than the similarity of the top and bottom layer of the same cluster. To avoid this, at the end of the clustering process MULICsoft can merge pairs of clusters whose top layers' modes' dissimilarity is less than the maximum layer depth of the two clusters. For this purpose, MULICsoft preserves the modes of the top layers of all clusters. Besides increasing the cluster coherence, this process reduces the total number of clusters, as well as the resulting error rates. This process is described as follows:

for (c = first cluster to last cluster)
  for (d = c+1 to last cluster)
    if the dissimilarity between c's mode and d's mode is less than the maximum layer depth of c and d, merge c into d and break the inner loop;

where the dissimilarity between two modes ($Q_c = \{q_{c1}, \ldots, q_{cm}\}$ and $Q_d = \{q_{d1}, \ldots, q_{dm}\}$) is defined as:

$$\text{dissimilarity}(Q_c, Q_d) = \sum_{i=1}^{m} d(q_{ci}, q_{di})$$

$$\text{where } \delta(q_{ci}, q_{di}) = \begin{cases} 0 & (q_{ci} = q_{di}); \\ 1 & (q_{ci} \neq q_{di}). \end{cases}$$

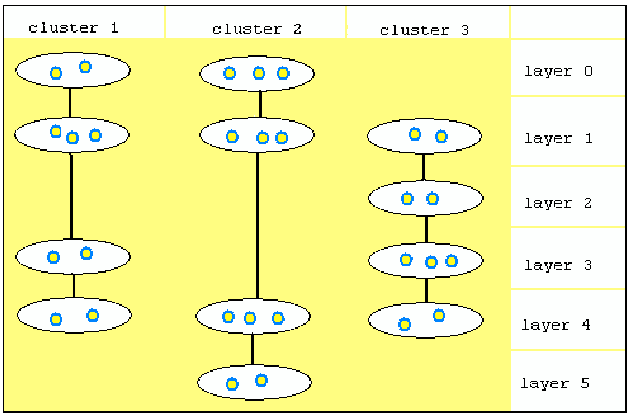

Fig. 3 –MULICsoft results. Each cluster consists of one or more different layers representing different coherences and similarities of the objects attached to the cluster.

4.2 Optimized Version of MULICsoft

We developed an optimized version of MULICsoft for runtime purposes. The optimized version increases the similarity criterion *num_values_can_differ* by 3 or 5 at a loop, while the non-optimized version increases it by 1 at a time. Sometimes - but not always - there is a slight loss in accuracy for the optimized version. On the other hand, sometimes there is a gain in accuracy. The runtime is significantly better. It is also possible to increase *num_values_can_differ* exponentially instead of linearly.

4.3 Dealing with Outliers

MULICsoft can eventually put all the objects in clusters. When *num_values_can_differ* equals the number of attributes, an unclassified object can be inserted in the lowest layer *h* of any existing cluster. This is undesirable if the object is an outlier and has little similarity with any cluster. The user can disallow this situation from happening, by specifying a maximum value for *num_values_can_differ* – represented as *threshold* in Figure 2 – therefore, when this value is reached, any remaining objects are not classified and are treated as outliers. As discussed in Section 5, the quality of the results often improves by treating the lowest-layer objects as outliers.

4.4 MULICsoft Extensions to MULIC

MULICsoft includes extensions to MULIC for network clustering. A reason why network categorical data needs a special clustering method is that the data contains mostly 'zeros' with 'ones' occurring sparsely.

A mode for a cluster contains 'zeros' by default. A position of the mode is set to 'one' if there is at least one object in the cluster that has a CA of 'one' in the corresponding position, or has a weight greater than zero at the corresponding position.

When calculating the similarity of a mode and an object, pairs of 'zero' attribute values between mode and object are ignored. The CAs in an object are represented as o_i.

All CAs in an object have "weights" in the range of 0.0 to 1.0 associated with them, which are inversely related to the number of unknown IPs in an indirect link between ASs. We represent an object's weights as *f_object*.

A novel similarity metric is defined to compute the similarity between a mode and an object, considering the CAs with their weights, as described next.

4.5 MULICsoft Similarity metric

A similarity metric is needed to choose the closest cluster to an object by computing the similarity between the cluster's mode and the object. We used the similarity metric described below, which considers the categorical attribute values and their weights. Our *f_similarity* metric amplifies the object positions having high weights, at pairs of CAs between an object *o* and a mode *m* that have identical values of 'one'. The weights of object o are represented as *f_object*.

$$f\_similarity(o, m) = \sum_{i=0}^{I} \frac{9 - (4 \times f\_objecti)}{5 - (4 \times f\_objecti)} \times \delta(oi, mi)$$

$$\text{where } \delta(o_i, m_i) = \begin{cases} 1 & (o_i = m_i = 1); \\ 0 & otherwise \end{cases}$$

Figure 4 shows the shape of the values returned by this similarity metric. Each object in this example has 10 CAs and weights. This graph shows that an object will be much more likely to be assigned to a cluster if all CAs match the mode with high weights of 1.0, than if all CAs match the mode with medium weights of 0.5, than if all CAs match the mode with low weights of 0.1, than if 1 CA matches the mode with a high weight, than if 1 CA matches the mode with a low weight.

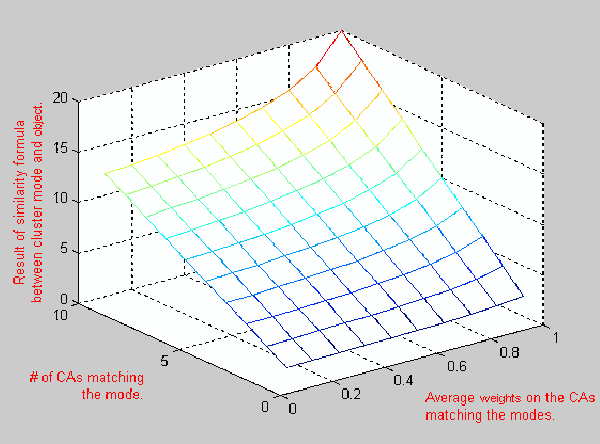

Fig. 4. The function surface of *f_similarity*, using values for the weights between 0.0 and 1.0 as described previously.

5. RESULTS FOR NETWORK DATA

We clustered the data on ASs that is provided by CAIDA. This data represents the links between ASs that form the Internet backbones [3-6, 15, 16].

For evaluating the accuracy of the results we followed an approach similar to that of using HA-values [14], by using the formula below. *a* represents the number of pairs of ASs that have at least 10 links between them (direct or indirect) and are in the same cluster, *b* represents the number of pairs of ASs that have at least 10 links between them and are in different clusters, *c* represents the number of pairs of ASs that do not have 10 links between them and are in the same cluster and *d* represents the number of pairs of ASs that do not have 10 links between them and are in different clusters:

$$\text{Quality of results} = \frac{a+d}{a+b+c+d}$$

We compared the results of MULICsoft with the results of k-Modes [13] and AutoClass [21] using the quality formula. The results are shown in Table 1 below. Based on our quality formula, we conclude that MULICsoft produced better results.

Table 1 – Quality of results for network data clustering

Clustering algorithm	Quality of results
MULICsoft	62%
k-Modes	40%
AutoClass	37%

5.1 Treating Objects as Outliers by Setting an Upper Bound for *num_values_can_differ*

We have found that by treating the last 500-1000 objects in bottom layers of clusters as outliers the quality improves. Objects are treated as outliers by setting an upper bound – represented as *threshold* in Figure 2 - for the variable *num_values_can_differ*. For example, setting a *threshold* for *num_values_can_differ* of 150 means that clustering will stop at layer 150 and any objects in layers greater than 150 will be treated as outliers. This supports the idea that lower layers are less reliable than higher layers. We have experimented with various *thresholds* for *num_values_can_differ*, for a linear growth of *num_values_can_differ* as discussed in Section 4.2. No merging is done on the clusters after the clustering.

Table 2 – Quality of results for different stop criteria.

threshold	Quality of results
100	61%
150	60%

6. M-VALUES FOR IDENTIFYING SIGNIFICANT CAs (ASs) IN A CLUSTER

Given a resulting cluster, we assigned a P1-value to each CA in the cluster; the term 'P1-value' was derived from the statistical 'P-value'. A P1-value measures whether a cluster contains a CA of a particular type more frequently than would be expected by chance. A P1-value close to 0.0 indicates a frequent occurrence of the CA in the cluster, while a P1-value close to 1.0 indicates its seldom occurrence. We multiplied the resulting P1-value with the reciprocal of the average of all weights assigned to the CA in the cluster, 1/avg(CV), thus resulting in what we call an *M-value*. M-values allow us to take into consideration the probability that a particular CA occurs in the cluster more frequently than expected by chance, in addition to the average weight of the CA in the cluster. For CAs that occur only once or twice in a cluster, a high P1-value results with an avg(CV) trivial to estimate [1].

The M-values were used for identifying in each cluster the CAs (ASs) with the lowest M-values. This gave us an indication of the most prominent ASs in a cluster. These ASs are very likely to be central points of communication in a cluster, acting as major links of communication for the ASs represented by the objects in the cluster. Table 3 shows some of the most significanct ASs identified in different clusters.

Table 3 – Most significant ASs identified in clusters.

Cluster id	AS id	# links in data set from/to AS	P-value	Avg(CV)
C20	AS209	700	0.01	1
C40	AS668	800	0.02	0.95
C60	AS3356	700	0.01	0.95

7. DISCUSSION AND CONCLUSION

An AS network data set is typically composed of many tight and highly coherent subgroups of ASs that through merging form larger less coherent groups. Although human identification of the major groups in a network data set is often impossible, identifying highly coherent subgroups of ASs is very helpful when trying to understand a large network data set. We have applied MULICsoft clustering on the AS network data set provided by CAIDA representing links between ASs. Figure 5 shows a graph of Internet topology, highlighting the tight clusters of connectivity. It is clear that there are many such tight clusters and they are often of a small size.

MULICsoft offers many advantages for clustering of AS network data. It focuses from the start on identifying all of the most coherent clusters as shown in Figure 5. MULICsoft finds all of the very coherent clusters that naturally exist in the data set. MULICsoft allows for a flexible number of clusters to be produced and does not sacrifice the coherence of the data sets for the number of clusters, which in k-Modes is defined strictly before the process. We showed that when requiring a strict number of clusters to be output from the clustering process, the resulting clusters might not have the maximum coherence [2].

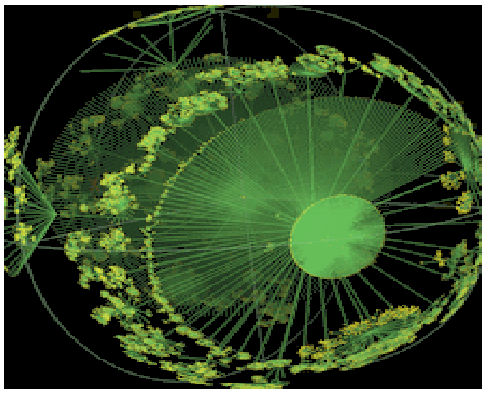

Fig. 5. A 3D hyperbolic graph of Internet topology. It was created using the Walrus visualisation tool developed by Young Hyun at CAIDA [5].

For each cluster, MULICsoft forms layers of varying coherence. It starts by forming a layer of high coherence using strict criteria concerning which objects to insert in the layer. As the process continues MULICsoft relaxes its criteria, forming layers that may be less coherent than the previous layers. A MULICsoft cluster is very representative of the underlying patterns in a network data set, as shown in Figure 5, because it recognizes that differing layers of coherence and similarity exist in a cluster amongst network objects.

Our method provides the user with the option to merge any clusters that are very similar, to build larger clusters and reduce the total number of clusters after the clustering process. We are currently implementing and testing an improved method for merging the clusters. This improved merging method will further improve the results of MULICsoft, because some times the results are too refined.

By incorporating weights on the CAs in the clustering process, one can use significance metrics considering the CAs and the weights, to identify the most prominent CAs in a cluster. These are the CAs that occur frequently and which exhibit high average weights in the cluster. Future work will include developing more significance metrics for the CAs and applying MULICsoft to more network data sets.

8. REFERENCES

[1] B. Andreopoulos, A. An and X. Wang. (2005) Clustering Mixed Numerical and Uncertain Categorical Data with M-BILCOM: Significance Metrics on a Yeast Example. Technical Report # CS-2005-03. Department of Computer Science, York University.

[2] B. Andreopoulos, A. An and X. Wang. (2004) MULIC: Multi-Layer Increasing Coherence Clustering of Categorical Data Sets. Technical Report # CS-2004-07. Department of Computer Science, York University.

[3] A.Broido, KC Claffy. Internet topology: Connectivity of IP graphs. SPIE 2001 conference on Scalability and Traffic Control in IP Networks Denver.

[4] A.Broido, KC Claffy. Complexity of global routing policies. CAIDA 2001.

[5] Cooperative Association for Internet Data Analysis (CAIDA). Data set ITDK0304. http://www.caida.org/

[6] H. Chang, R. Govindan, S. Jamin, S. Shenker, and W. Willinger, "Towards capturing representative AS-level Internet topologies", Technical Report UM-CSE-TR-454-02, University of Michigan Computer Science, 2002.

[7] J. Chen and Lj. Trajkovic, Analysis of Internet topology data, Proc. IEEE Int. Symp. Circuits and Systems, Vancouver, British Columbia, May 2004, vol. IV, pp. 629-632.

[8] K.C. Claffy. (1999) Internet measurements and data analysis: topology, workload, performance and routing statistics. NAE'99 workshop.

[9] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the Internet topology, In Proc. of ACM SIGCOMM '99, Cambridge, MA, Aug. 1999, pp. 251-262.

[10] Goebel, M. & Gruenwald, Le (1999). A survey of data mining and knowledge discovery software tools. ACM SIGKDD Explorations 1, 20-33.

[11] Grambeier J., Rudolph A. (2002) Techniques of Cluster Algorithms in Data Mining. Data Mining and Knowledge Discovery 6: 303-360.

[12] Hartigan, J. A. Clustering algorithms. (John Wiley and Sons, New York, 1975).

[13] Huang Z. (1998) Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values. Data Mining and Knowledge Discovery 2(3): 283-304.

[14] L. Hubert and P. Arabie, "Comparing partitions", Journal of Classification, 193–218, 1985.

[15] B.Huffaker, A.Broido, KC Claffy, M.Fomenkov, K.Keys, Y.Hyun, D.Moore. Skitter AS Internet Graph. CAIDA, April-May 2003.

[16] B.Huffaker, D. Plummer, and D. Moore, "Topology discovery by active probing". CAIDA, 2002.

[17] Yun Li, H.S. Venter, J.H.P. Eloff. Categorizing vulnerabilities using data clustering techniques. (2004) Information Security South Africa (ISSA 2004).

[18] M. Mihail, C. Gkantsidis, and E. Zegura, Spectral analysis of Internet topologies, Proc. Of Infocom 2003, San Fancisco, CA, March 2003, vol. 1, pp. 364-374.

[19] Leonid Portnoy, Eleazar Eskin and Sal Stolfo. Intrusion Detection with Unlabeled Data Using Clustering (2001).ACM Workshop on Data Mining Applied to Security (DMSA 01).

[20] Solka, J. L. and Marchette, D. J. (2001), "Functional Analysis of Computer Network Data", *Computing Science and Statistics*, 33, /I2001Proceedings/JSolka/JSolka.pdf

[21] Stutz J. and Cheeseman P. (1995) Bayesian Classification (AutoClass): Theory and results. Advances in Knowledge Discovery and Data Mining, 153-180, Menlo Park, CA, AAAI Press.

[22] D. Vukadinovic, P. Huang, and T. Erlebach, On the Spectrum and Structure of Internet Topology Graphs. In proceedings of I2CS, June 2002, Kühlungsborn, Germany.

[23] Nong Ye, Xiangyang Li. A Scalable Clustering Technique for Intrusion Signature Recognition, Proceedings of the 2001 IEEE Workshop on Information Assurance and Security United States Military Academy, NY, June 2001.

[24] Wenke Lee and Salvatore Stolfo. Data mining approaches for intrusion detection, Proceedings of the 7th USENIX Security Symposium, San Antonio, TX, 1998.