# P5: Anomaly Detection Project Report

## Algorithm

Study of structural similarities of graphs can be useful in plethora of applications like clustering, anomaly detection, regenerating the graph nodes and edges which are missing from structurally similar graphs, etc. However,the task of determining the similarity was being done using node correspondence. Netsimile proposes an algorithm which does not use node correspondence to identify the similarity in the graphs. It is based on the idea of representing a graph with a signature, which identifies the structural properties of the graph. Representing the graph by using a signature vector has following advantages over other methods:

1) It is size invariant
2) The algorithm is scalable
3) No need to solve node-correspondence problem.

The algorithm has three steps:

- Feature extraction
- Feature aggregation
- Similarity comparison

**Feature extraction:**

In this step the features of graph are extracted. Different features that are extracted are as follows.

- di = degree of node i
- ci = clustering coefficient of node i. Clustering coefficient is the number of triangles connected to node I divided by the number of connected triplets centered on node i.
- d (N (i)): average number of node i's two-hop away neighbors. Sum to two hop away neighbors of node divided by the degree of node.
- C (N (i)): Average clustering coefficient of node i. Sum of clustering indices of neighbors of node i divided by degree of node.
- |E (ego (i))|: number of edges in node i's ego net. Ego net is the sub graph induced by the node and its neighbors.
- |Eo(ego(i)|: number of outgoing edges from eog (i).
- |N (ego (i))|: number of neighbors of ego (i).

All the above mentioned features are extracted for each node of the graph. Resulting into node * 7 matrix since number of features is 7.

**Feature aggregation:**

In the feature aggregation step, the features are aggregated using the following metric

Mean of each feature, Median of each feature, standard deviation of each feature, skewness of each feature and kurtosis of each feature. This results into a vector of size 35 where the aggregator of each feature are placed one after other. Since there are 5 metrics for each feature and there are in all 7 features, we have 5*7 = 35 numbers which represent a graph.

This vector serves as a signature for the graph.

**Similarity Comparison:**

Once the signature of the graph is obtained it can be used for clustering, anomaly detection, similarity calculation between two graphs etc. using the distance between the signature vectors.

Canberra distance proved to be the most effective one to determine the similarity between two signature vectors.

**Anomaly detection in time evolving graphs using NetSimile:**

In order to detect anomalies in time evolving graphs, Canberra distance is calculated between consecutive graphs. After that, threshold is calculated using the following formula:

**Threshold calculation Method 1;**

MR = $|X1-X2| + |X2-X3| + \text{------} + |X_{k-1} - X_k| / k-1$

Threshold is: median + 3 * MR.

**Threshold calculation Method 2:**

I also used one more method to calculate the threshold. The method is as follows:

n = number of measurements in moving window = 2

MR = |Current distance – previous distance|

Rbar = mean of MR

Xbar = mean of distances

$\sigma c$ = Rbar / 1.128

Upper Control Limit = Xbar + 3 * $\sigma c$/sqrt(n)

The above method can be useful to determine anomalies by setting the constant c to an appropriate value.

**Anomalies:**

G(t) refers to the graph at time t.

Now if the distance between Graph G(t) and G(t+1) is above threshold and the distance between graph G(t+1) and G(t+2) is also above threshold, then graph G(t+1) is an anomaly since it is causing the similarity measure to cross threshold.

## Parameters impacting the Algorithm

The two parameters which affect the algorithm results are:

- Threshold
- Distance metric for similarity calculation

The threshold can be calculated by different methods, two of which I described above. Depending on the threshold the output of the program is different. Which threshold produces optimum output can be understood if we have the ground truth. In absence of ground truth, the accuracy cannot be verified.

The distance metric used to calculate the similarities can also impact the results. As mentioned in the paper, Canberra distance performs the best. So I used Canberra distance as a metric to determine similarity.

## Time Complexity

**Feature Extraction**: requires O (f*n) time as each feature value calculation requires O(n) time.

**Feature aggregation:** requires O (f*nlog (n)) time. Median requires the values to be sorted which takes nlogn time where n is the number of nodes in the graph. Rest other features can be calculated in O (n) time. Thus the overall complexity is O(f * n log(n)).

**Similarity Comparison:** Similarity calculation takes constant time as it is just a distance calculation step. The length of signature vector is 35 which is constant and hence constant time.

## Algorithm Performance

This algorithm performs the best in comparison to other algorithms for graph anomaly detection or event change detection.

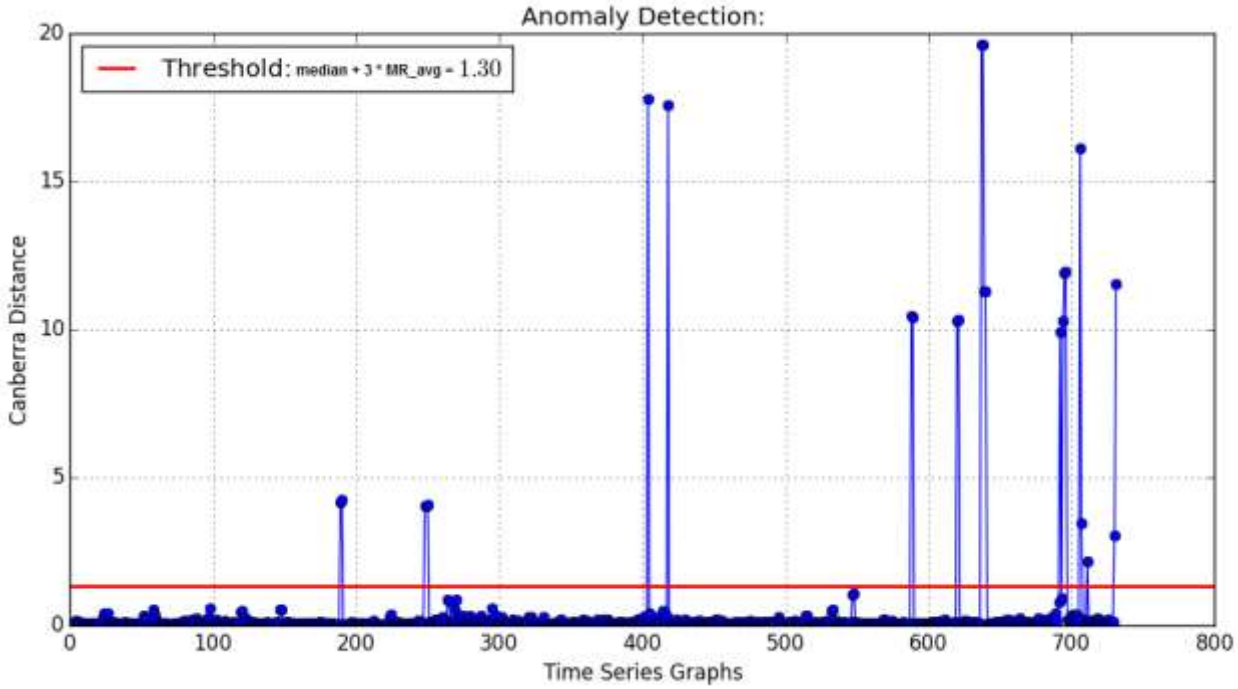
This algorithm is highly scalable as it doesn't depend on the size of the graph and robust also. Only improvement required in the algorithm is appropriate threshold calculation method based on problem type.

**This algorithm can only detect graph level anomalies. Node level, edge level and subgraph level anomalies cannot be determined by the algorithm.**

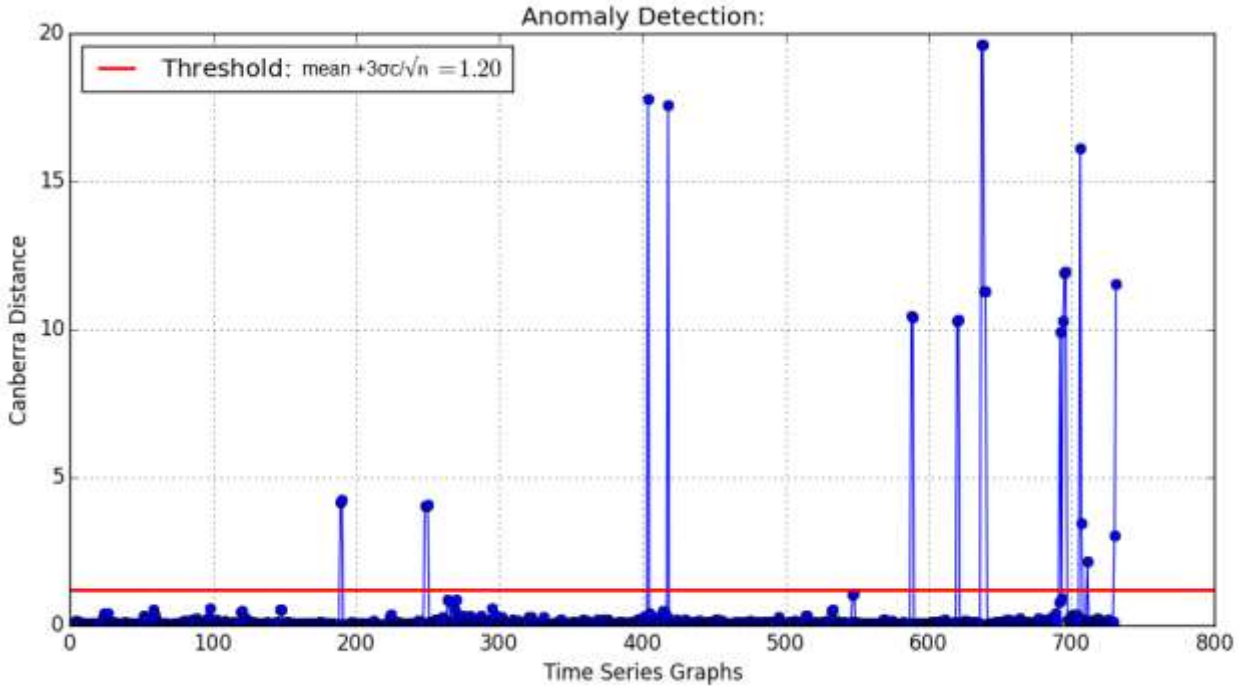
This algorithm is not randomized.

## Program Output

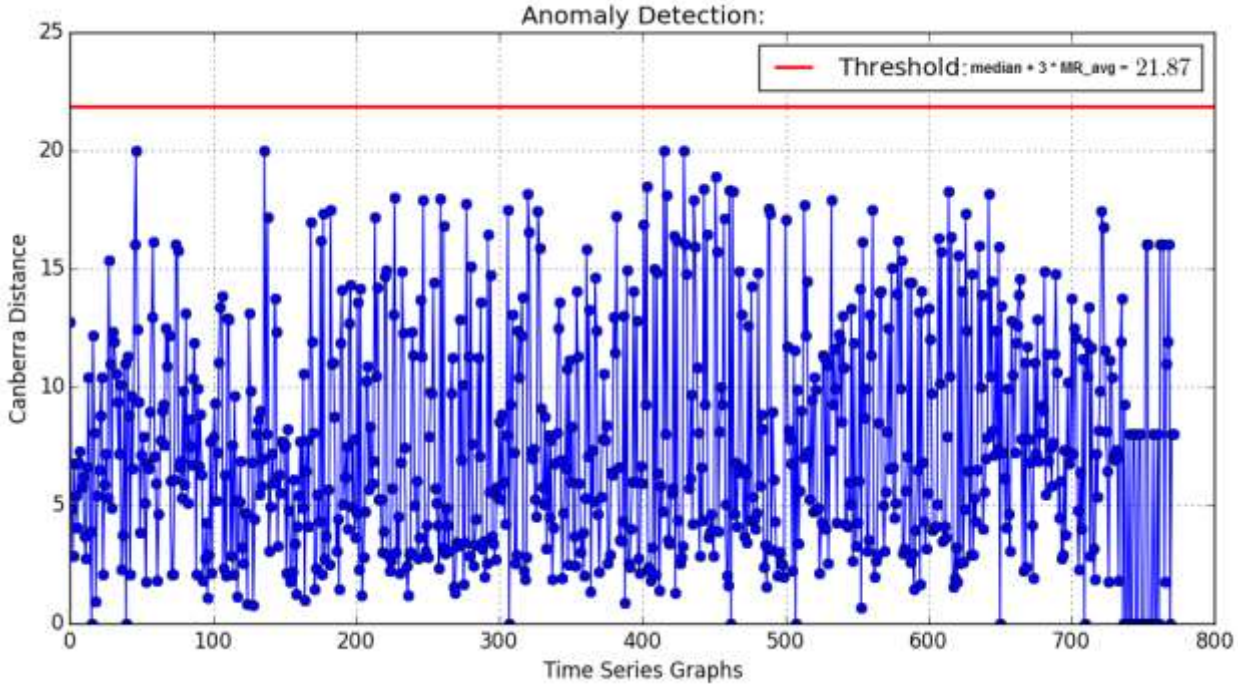
The output of the program for as-733 graphs by using the threshold calculated using method 1 is as follows:



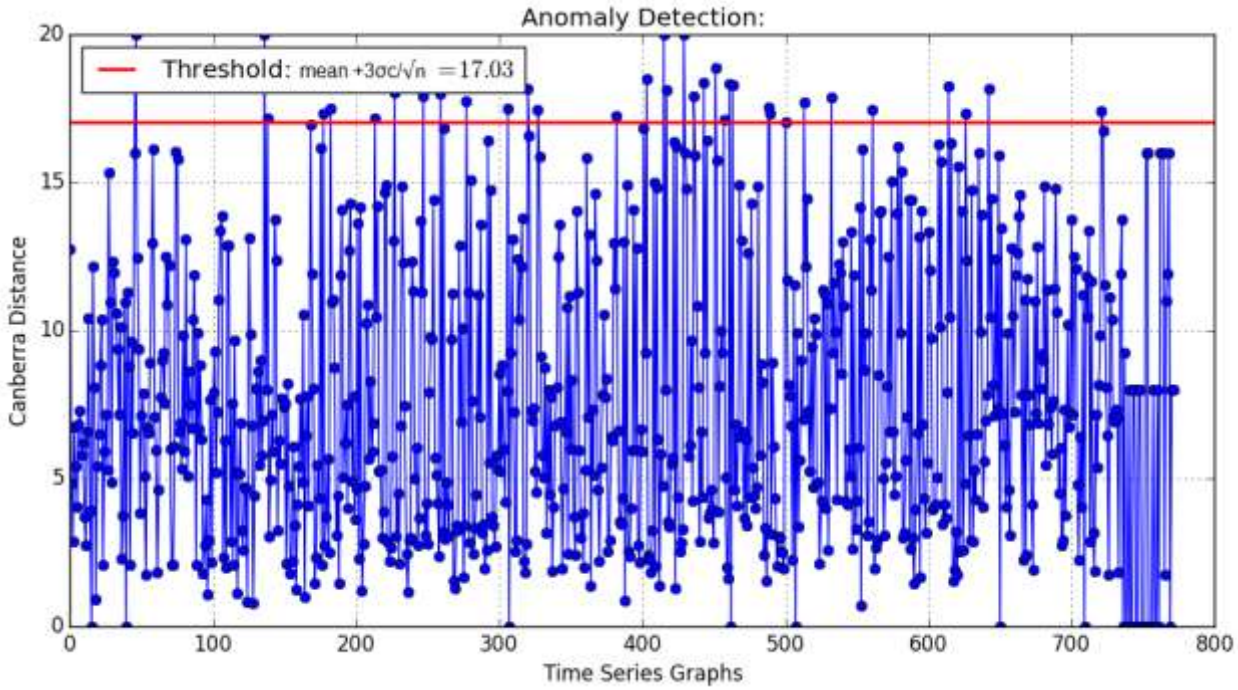

The output of the program for as-733 graphs by using the threshold calculated using method 2 is as follows:



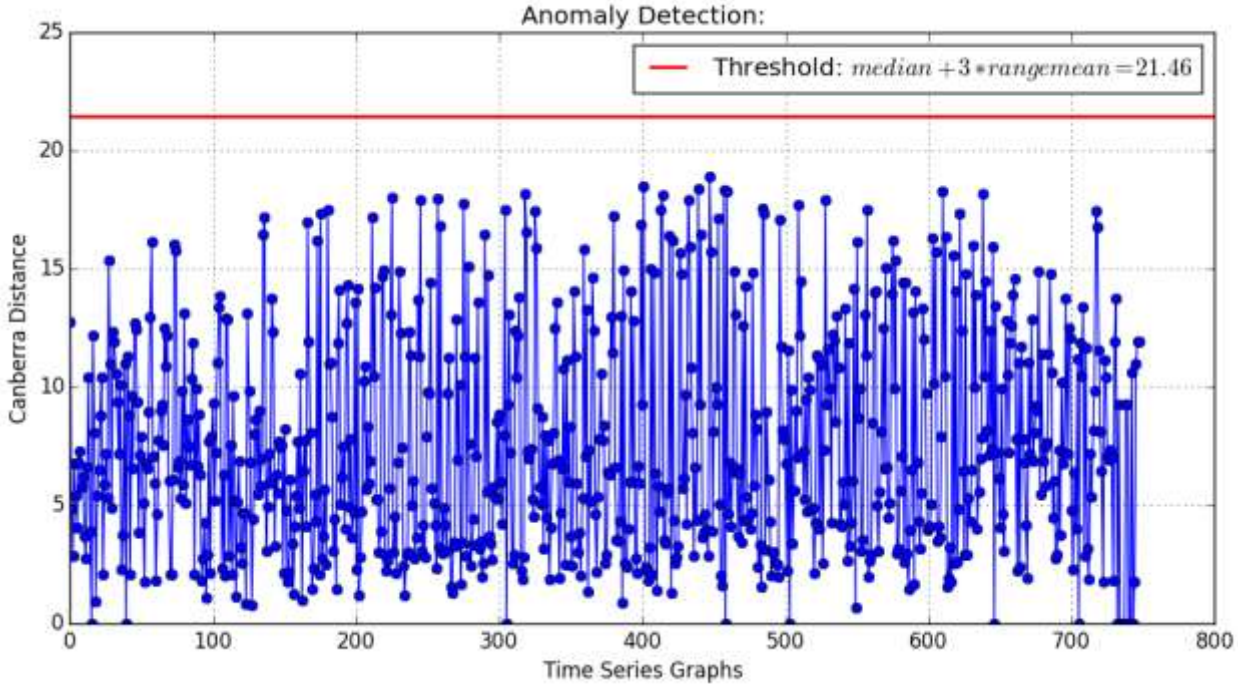
The output of the program for Enron graphs by using the threshold calculated using method 1 is as follows:



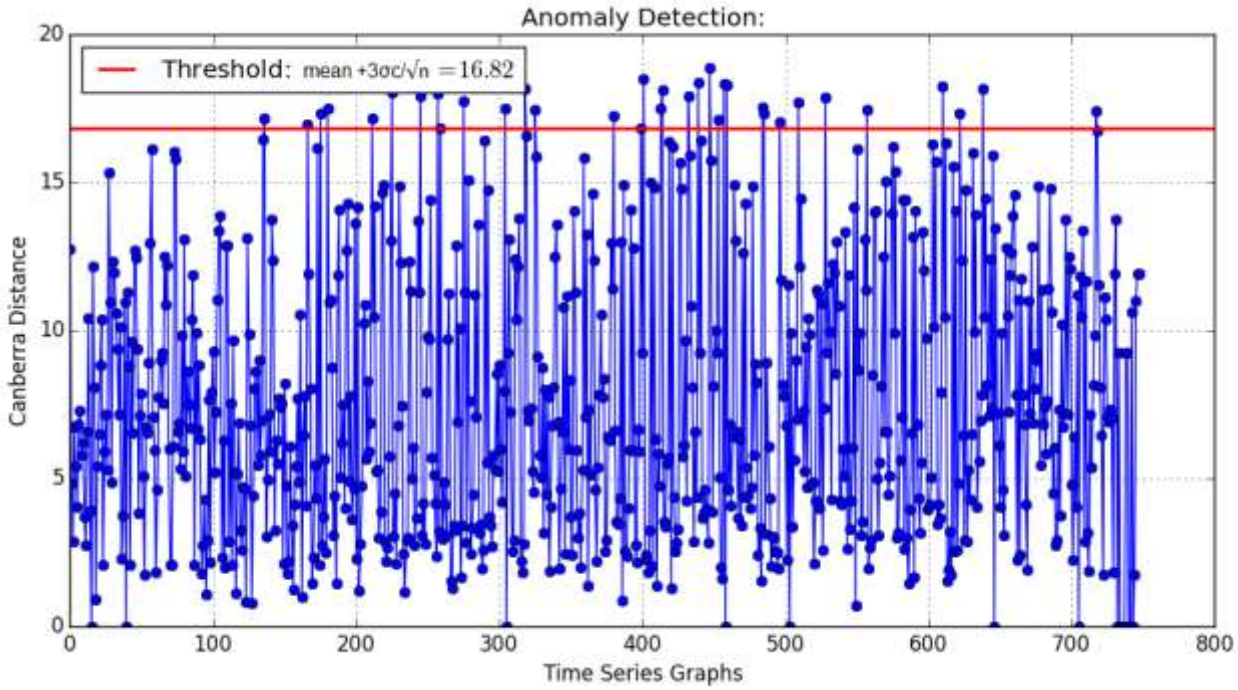

The output of the program for Enron graphs by using the threshold calculated using method 1 is as follows:



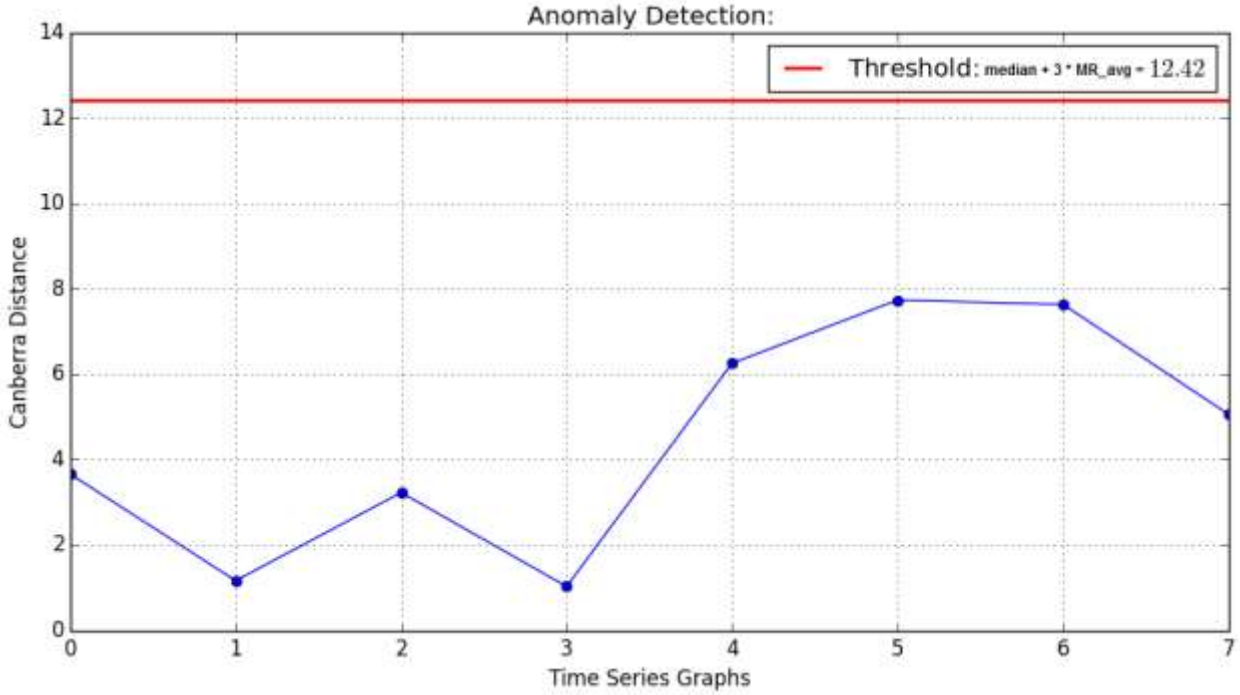
The output of the program for Enron-nonempty graphs by using the threshold calculated using method 1 is as follows:



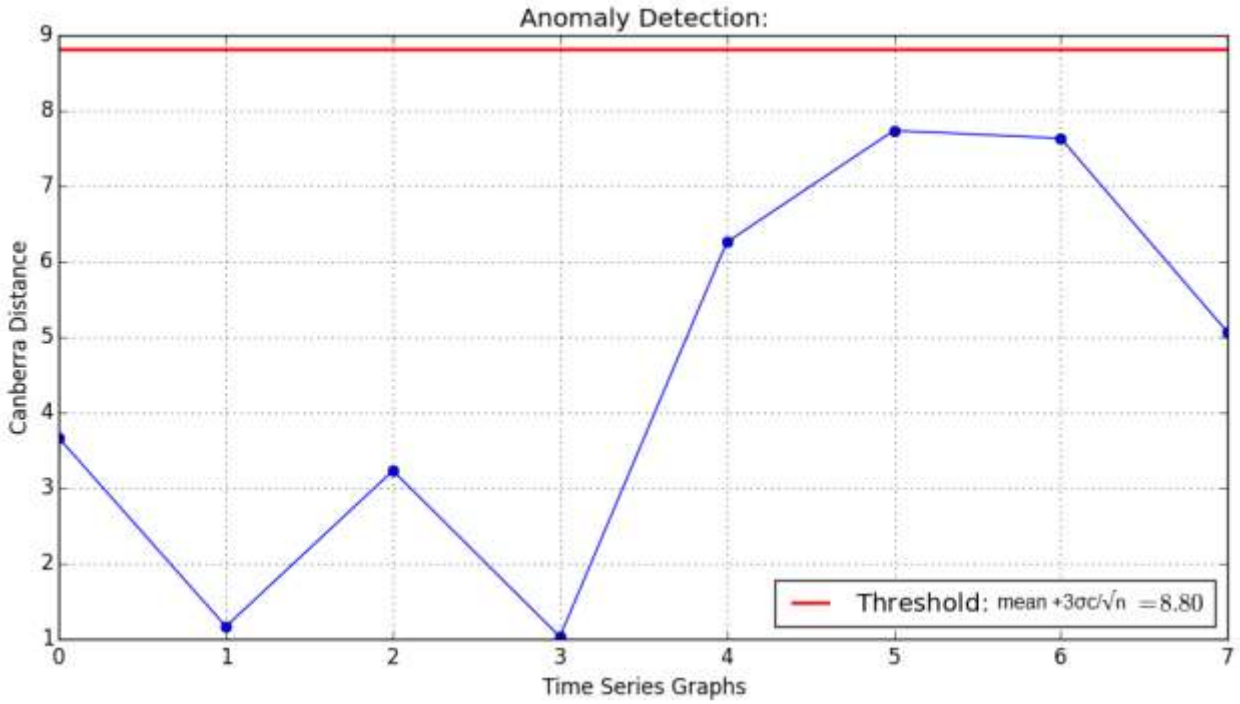

The output of the program for Enron-nonempty graphs by using the threshold calculated using method 2 is as follows:



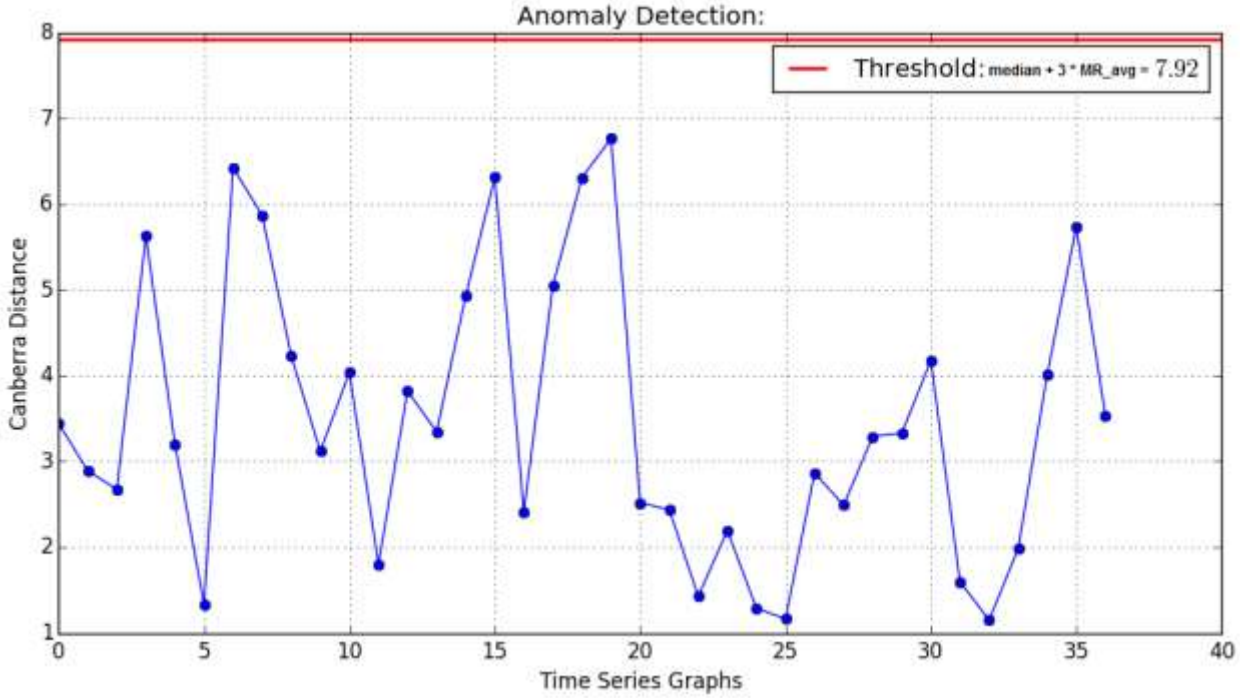
The output of the program for Gnutella graphs by using the threshold calculated using method 1 is as follows:



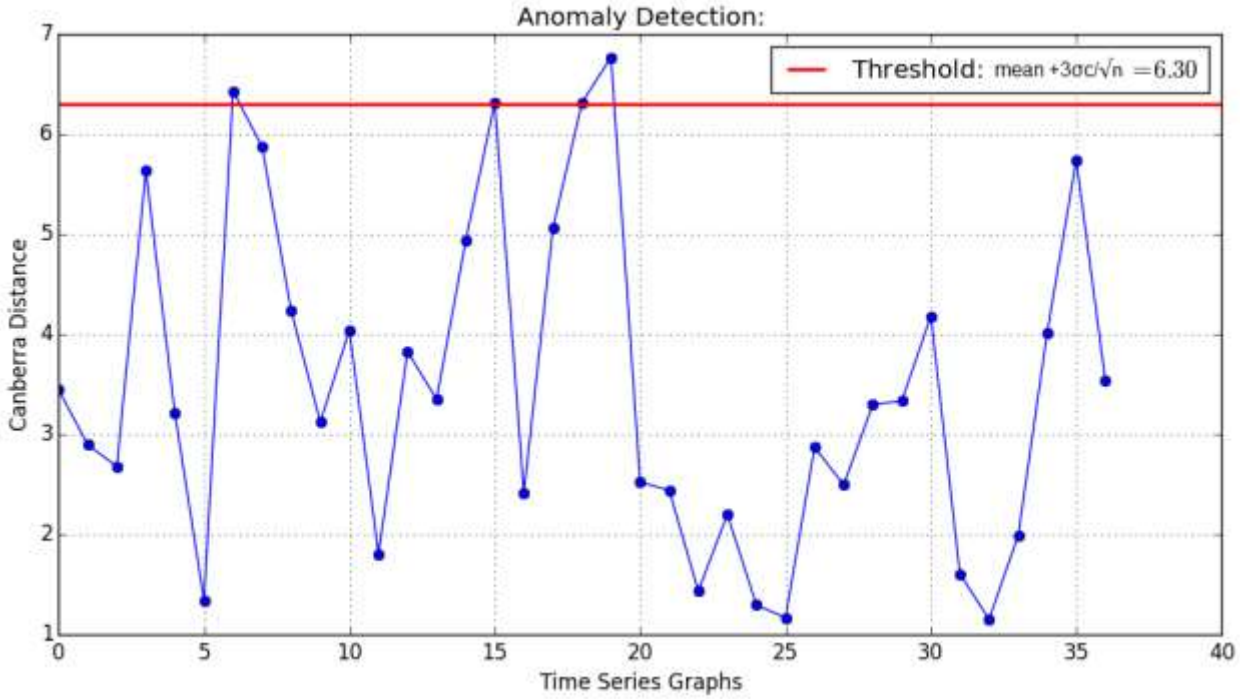

The output of the program for Gnutella graphs by using the threshold calculated using method 2 is as follows:



The output of the program for Reality Voices graphs by using the threshold calculated using method 1 is as follows:




The output of the program for Reality Voices graphs by using the threshold calculated using method 2 is as follows:



## Discussions on Results and Algorithm Shortcomings

The algorithm performed well on all graph sets without running out of memory.

The time taken by the algorithm for as-733 graph dataset was around 3:00 hours while for Gnutella it took 20 minutes. For rest other graph datasets, the program executed in seconds.

Anomalies were not detected on any graph datasets **using threshold calculated by method 1**. For As-733, 12 anomalies were detected from 733 graphs, which is 1.6% of total graphs were anomalous.

**Using threshold calculated by method 2,**

12 anomalies were found in as-733 graphs which is 1.6% of total number of graphs.

1 anomaly was found in enron graph dataset from 733 graphs which is 0.13% of total number of graphs.

2 anomalies were found in enron-non empty graph dataset from 749 graphs which is 0.26% of total number of graphs.

1 anomaly was found in reality mining voices graph dataset from 38 graphs which is 2.6% of total number of graphs.

Thus the sensitivity of the algorithm to the outliers depends on threshold. Method 1 is less sensitive to outliers as compared to method 2. The sensitivity of algorithm depends on threshold calculation method.

The method proposed will never output graph at 1$^{st}$ time point and graph at last last time point as anomalous as they cannot be compared with previous and next time points.

There is no relation between the spread of anomalous time points and the algorithm. Some anomalous time points are spread out and some are near each other.

If we see the anomalous time point 190 in as733 by taking threshold obtained by method 1, then the number of edges in the graph suddenly reduced by 1400+ edges in comparison to nearby graphs. Due to this, the distance between graph structure at time points 189 & 190, 190 &191 increases.