Article

An improved data clustering algorithm for outlier detection

Anant Agarwal, **Arun Solanki**
Department of Computer Science and Engineering, Gautam Buddha University, Greater Noida, India
E-mail: anantagarwal93@gmail.com, ymca.arun@gmail.com

Received 4 May 2016; *Accepted 12 June 2016*; *Published 1 December 2016*

Abstract
Data mining is the extraction of hidden predictive information from large databases. This is a technology with potential to study and analyze useful information present in data. Data objects which do not usually fit into the general behavior of the data are termed as outliers. Outlier Detection in databases has numerous applications such as fraud detection, customized marketing, and the search for terrorism. By definition, outliers are rare occurrences and hence represent a small portion of the data. However, the use of Outlier Detection for various purposes is not an easy task. This research proposes a modified PAM for detecting outliers. The proposed technique has been implemented in JAVA. The results produced by the proposed technique are found better than existing technique in terms of outliers detected and time complexity.

Keywords data mining; outlier detection; clustering; pam; java.

Selforganizology
ISSN 2410-0080
URL: http://www.iaees.org/publications/journals/selforganizology/online-version.asp
RSS: http://www.iaees.org/publications/journals/selforganizology/rss.xml
E-mail: selforganizology@iaees.org
Editor-in-Chief: WenJun Zhang
Publisher: International Academy of Ecology and Environmental Sciences

1 Introduction

Data mining is the process of extracting previously unknown useful information from data (Padhy et al., 2012). Data mining is a powerful concept for data analysis and discovering patterns from a huge amount of data (Smita and Sharma, 2014). Because of data mining, important knowledge is derived from the collection of data. Data mining is mostly used for the purpose of assisting in the analysis of observations (Vijayarani and Nithya, 2011). Data mining techniques make use of collected data to build predictive models (Jain and Srivastava, 2013).

Clustering is a typical methodology for grouping similar data together (Zhang et al., 2014; Zhang, 2016; Zhang and Li, 2016). A cluster is a set of data objects which are similar to each other data object present in the same cluster and are dissimilar to data objects in other clusters (Rajagopal, 2011). Clustering is one of the most fundamental operation of data mining. A good clustering algorithm can identify clusters regardless of the shapes (Kaur and Mann, 2013). Various different types of clustering algorithms exist like K-Means (Yadav and Sharma, 2013), DBSCAN (Vaghela and Maitry, 2014), EM (Nigam et al., 2011), etc. Each with its own novel set of processes that lead to the clustering of the data.

Outlier detection is a branch of data mining which has many applications. The process of detecting outliers

is very important in data mining (Mansur et al., 2005). Outliers are basically points that do not conform to the general characteristic of the data. Outlier Detection aims to find patterns in data that do not conform to the expected behavior (Singh and Upadhyaya, 2012). There are numerous techniques that can perform the task of detecting outliers and usually selecting which one to use proves to be the biggest challenge (Vijayarani and Nithya, 2011).

2 Literature Survey

Zhang et al. (2012) proposed an improved PAM clustering algorithm which involved the calculation of the initial medoids based on a distance measure. Points that were closer to each other were put in separate sets and the points closest to the arithmetic means of the sets were chosen as initial medoids. (Bo et al., 2012) made use of a minimal spanning tree for the pretreatment of the data. A minimal spanning tree of the data was constructed and then split to obtain k subtrees which resulted in the formation of k clusters. Paterlini et al. (2011) proposed a novel technique for initial medoid selection by using pivots. The approach is based on the intuition that a medoid is more likely to occur on a line joining two most distant objects.

Vijayarani and Nithya (2011) proposed a new partition based clustering algorithm for the purpose of outlier detection. The work was centered around the concept of selecting most distant elements as the initial medoids. Loureiro et al. (2004) described a method that uses the application of hierarchical clustering in the process of detecting outliers. Aggarwal and Kaur (2013) presented an analysis on the various partition based clustering algorithms utilized for outlier detection.

Al-Zoubi (2009) proposed a different technique for outlier detection in which clusters that have less than a specified number of data points are considered entirely as outlier clusters. The remaining outliers were identified by using a threshold value from the cluster center to each of the cluster members. Those data points that were beyond this threshold were detected as outliers. Karmaker and Rahman (2009) proposed a technique of outlier detection that can be applied to spatial databases. Duraj and Zakrzewska (2014) presented an outlier detection technique based on the simultaneous indication of outlier values by three well known algorithms: DBSCAN, CLARANS and COF.

3 System Architecture

As can be seen in Fig. 1, the architecture of the proposed work is a 3-Tier architecture with a Graphical User Interface (GUI), a Central Processing Unit and a Database.

3.1 Graphical User Interface (GUI)

The user is provided with a GUI to provide the necessary inputs to the system. This GUI also acts as the medium where the necessary results are displayed. The user is required to enter three values for the selection of dataset and one value for deciding the clustering criterion.

3.2 Database

The database used in this system is a dynamic database. The database is designed at runtime based on the values entered by the user. The database is displayed on the GUI as soon as the user enters the necessary values.

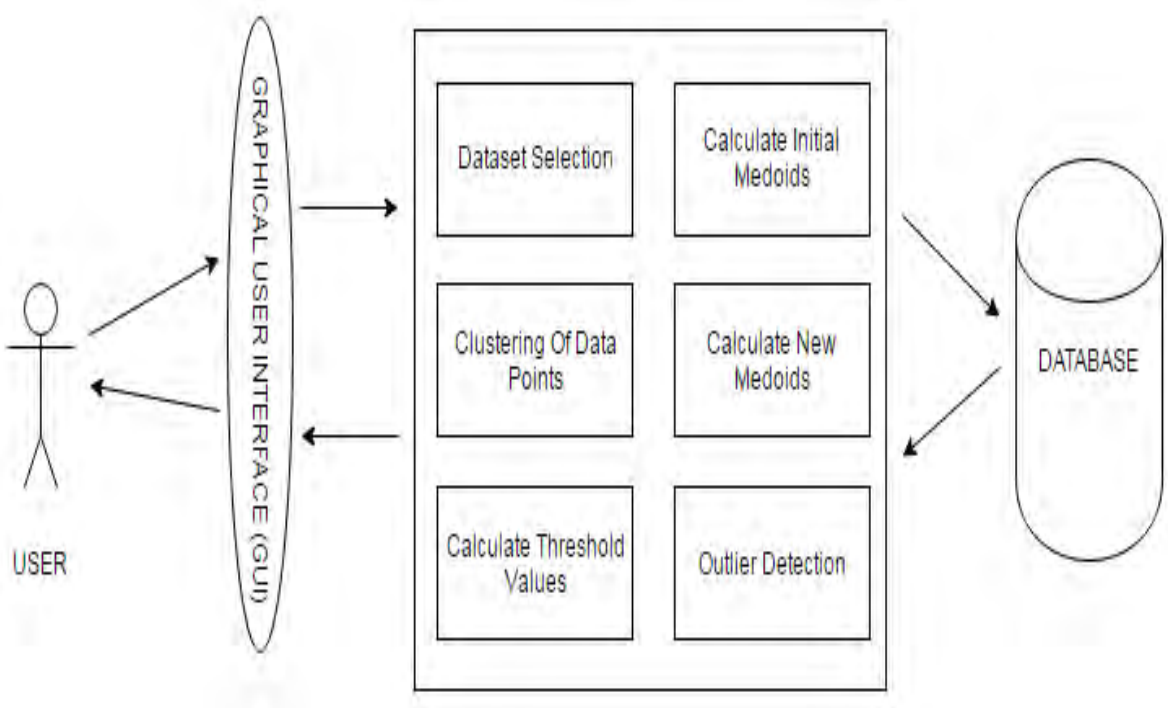


Fig. 1 System architecture of proposed work.

3.3 Dataset selection

This module takes three values as inputs and designs a dataset based on the three values. This module provides the design of the dataset that is used for the execution of the proposed technique and is responsible for storing it in the database.

3.4 Calculate initial medoids

This module takes the number of clusters (k) as input. It is responsible for the identification of k initial points. These identified points are taken as the initial medoids for the clustering process.

3.5 Clustering of data points

This module is responsible for assigning each data point to a cluster. This module takes the set of initial medoids as input and assigns each data point to the cluster represented by the closest medoid.

3.6 Calculate new medoids

This module takes the obtained clustering as input. It then starts from the initial medoids and selects that element in each cluster that minimizes the cost of the cluster configuration. Thus new medoids are obtained for each cluster such that the cost of the cluster configuration is minimized.

3.7 Calculate threshold values

This module takes the updated medoids as input. It then calculates the ADMP (Absolute Distance between Mean and Point) values for each cluster. It calculates the distance of the data points to their respective medoids. A threshold is calculated as 1.5 x (average ADMP in each cluster) for each cluster.

3.8 Outlier detection

This module takes the threshold values as input. This module identifies those data points which lie beyond the threshold distance from the medoid of its cluster and marks them as outliers.

4 Methodology of Proposed System

The traditional PAM algorithm involves the random selection of initial medoids. These initial medoids are then considered for achieving the clustering of the data. After the clustering has been done, new medoids are identified based on minimizing the cluster configuration cost for each cluster. Thus, a new set of medoids is obtained. These new medoids are then used to compute the distance between each non-medoid element and the medoid of its cluster. The average distances obtained for each cluster are then multiplied by a multiplication factor of 1.5 to obtain a threshold value for each cluster. All those data points that lie at a distance greater than the threshold from their respective medoids are termed as outliers. Elements inside the threshold are termed as inliers. Thus, it can be seen that the final clustering achieved is heavily dependent on the choice of initial medoids. As a result, the outlier detection phase is also dependent on the choice of initial medoids.

The proposed algorithm aims to choose a proper measure for obtaining the initial medoids. This way the algorithm doesn't suffer from the inherent blindness and randomness in initial medoid identification as is the case with the traditional PAM. The algorithm preserves the other processes of the PAM algorithm. The algorithm sorts the obtained data points in their ascending order of distance from the origin. It then calculates a factor which is equal to the ratio of the number of data points to the number of clusters required. The algorithm then selects the first factor number of elements from the dataset, removes them from the dataset and calculates the mean value of the extracted data points. An extracted data point is identified whose distance from the obtained mean is the smallest among all the extracted data points. This selected point is chosen to be a medoid. This process is repeated until the desired numbers of initial medoids have been obtained. After that, the algorithm proceeds to the clustering of the data, which is followed by the identification of new medoids. Then the thresholds are calculated and elements are identified as inliers or outliers for each cluster.

The distance metric used in the system is the Euclidean Distance. The Euclidean Distance between two points X (x_1, x_2……x_n) and Y (y_1,y_2….y_n) is:

$$d_E(X,Y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

5 Process Flow of Proposed Work

The process flow of the proposed system has been depicted in Fig. 2. The system consists of various different modules that interact with each other to complete the working of the system. The system begins by prompting the user to enter certain inputs. After these inputs have been provided by the user, the system initiates the necessary modules and then provides the user with the final results.

Step 1: The input module is the first module that is initiated when the system is executed. The module requires the user to enter four inputs. The first input requires the user to enter the number of data points (dp) on which the system needs to be executed. The second input requires the user to enter the number of clusters for the clustering process (k). The third and fourth inputs require the user to enter the largest allowed value for the x (x_{max}) and y (y_{max}) co-ordinates of the data points that are generated.

Step 2: The dataset selection module takes the values of dp, x_{max} and y_{max} as input. This module then designs a dataset containing dp data points such that all data points lie in the region bounded by (1,1) and (x_{max},y_{max}).

Step 3: The database takes the dataset design from the dataset selection module and loads a dataset into the memory. After the database has been created the system then starts the execution of the proposed algorithm.

Step 4: The medoid selection module takes the number of clusters (k) as input. It calculates k number of initial medoids that will be used later in the system. It calculates a factor as (dp/k). The module sorts the data according to the ascending distance of the data points from the origin. It then extracts the first factor number of

elements from the dataset, removes them from the dataset and calculates the mean point of the extracted points. The extracted point that is closest to the mean is chosen as a medoid. This process is repeated until k such medoids have been obtained.

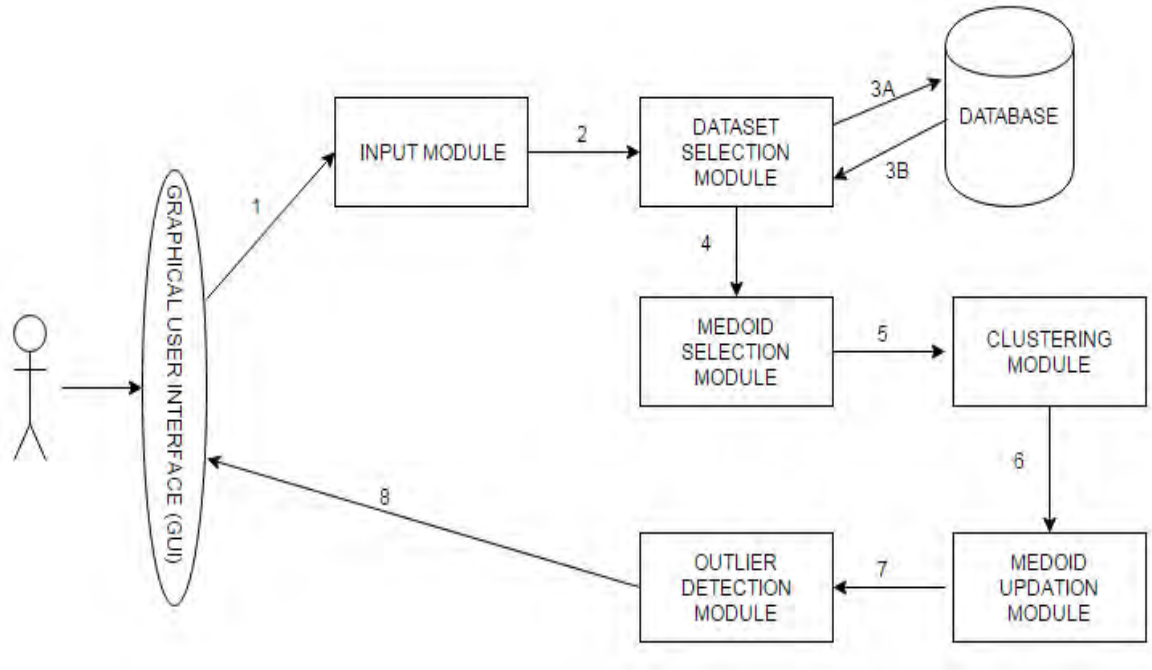


Fig. 2 Process flow diagram of proposed work.

Step 5: The clustering module takes the set of initial medoids as input. It then checks the distance between each data point and the previously obtained medoids. Based on the calculated distances the data point is assigned to the closest medoid, thus every data point is assigned to be in the cluster represented by the closest medoid.

Step 6: The medoid updation module takes the clustering obtained from the clustering module as input. It then selects a cluster and the medoid representing that cluster. The module calculates the distance between the medoid and each non-medoid point. The module then swaps the medoid point with a non-medoid point and recomputes the distances. If the latter distances are less than the former then the swap is kept, otherwise the swap is cancelled. The procedure is repeated for each cluster until there is no more shifting in the medoid point.

Step 7: The outlier detection module takes the updated medoids as input. The module then computes the distances between the medoids and the non medoid elements in each cluster. The distances are calculated using Euclidean distance formula. Once the distances have been evaluated, an average of the distance values is obtained for each cluster. A threshold is calculated by using a multiplication factor of 1.5 with the average value obtained. This is the threshold value for the detection of outliers. The distance of each data point is now checked with its corresponding medoid. If the distance is found to be larger than the threshold then the point is designated as an outlier. If the distance is found to be less than or equal to the threshold then the point is designated as an inlier.

Step 8: The final number of outliers detected by the proposed algorithm is displayed on the screen. The execution time of the algorithm is also displayed.

6 Pseudocode of Proposed Work

The pseudocode of the proposed work is as follows:

Step 1: Input number of clusters (k) and select data (dp).
Step 2: Sort data points according to the distance from origin and calculate factor as (dp/k).
Step 3: Choose the first (dp/k) elements of the dataset, remove them and calculate the mean of these elements.
Step 4: Select that data point from these elements which is the closest to the obtained mean and select it as a medoid.
Step 5: Repeat Steps 3 to 4 until k such elements have been identified.
Step 6: Select these k elements as medoids.
Step 7: Calculate the distances between each data point and the medoids. Associate each data point with the medoid for which it has the minimum distance. The data point becomes a member of the cluster that is represented by that medoid.
Step 8: Compute new medoids for each cluster. Consider a non-medoid data point d_n and calculate cost of swapping the current medoid d_m of the cluster to selected non-medoid data point d_n.
Step 9: If cost<0, set $d_m = d_n$.
Step 10: Repeat Steps 8 and 9 until no change.

7 Implementation and Working of System

The internal working of the system is discussed in this section. In Section 7.1, the implementation details of the system have been discussed. In Section 7.2, the working of the system is discussed in detail.

7.1 Implementation

The system has been implemented with the help of JAVA. Both algorithms i.e. the already existing PAM algorithm and the proposed algorithm have been implemented in JAVA. The tool used for JAVA is the NetBeans IDE (Integrated Development Environment). The GUI for the software is also JAVA based which has been implemented with the help of JAVA Swing tools and JFreeChart.

7.2 Working

This section explains the functioning of the designed system with the help of actual snapshots of the running system.

7.2.1 Homepage

As shown in Fig. 3, the homepage of the designed system consists of four text fields that need to be filled by the user. The first field requires the input of the number of data points to which the designed system needs to be applied. The second field takes the number of clusters that need to be taken during the clustering phase. The third field sets a range on the x-coordinate values of the generated points, such that all points lie in the region between 0 and the entered value. The fourth field sets a range on the y-coordinate values of the generated points, such that all points lie in the region between 0 and the entered value. The user can press OK after entering all the details to initiate the system. The user also has the option to press the CANCEL button and terminate the system. In this case the user has entered values such that 1000 data points have to be generated in the region (1,1) to (100,100). The number of clusters for clustering is chosen as 4.

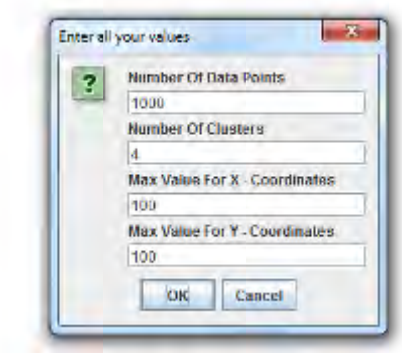


Fig. 3 Homepage.

7.2.2 Load dataset

As shown in Fig. 4, 1000 data points have been generated such that every point occurs in the region from (1,1) to (100,100). Each data point is denoted by a red square on the graph. It can be seen that the data points obtained by the system are completely in compliance with the values that were provided by the user initially.

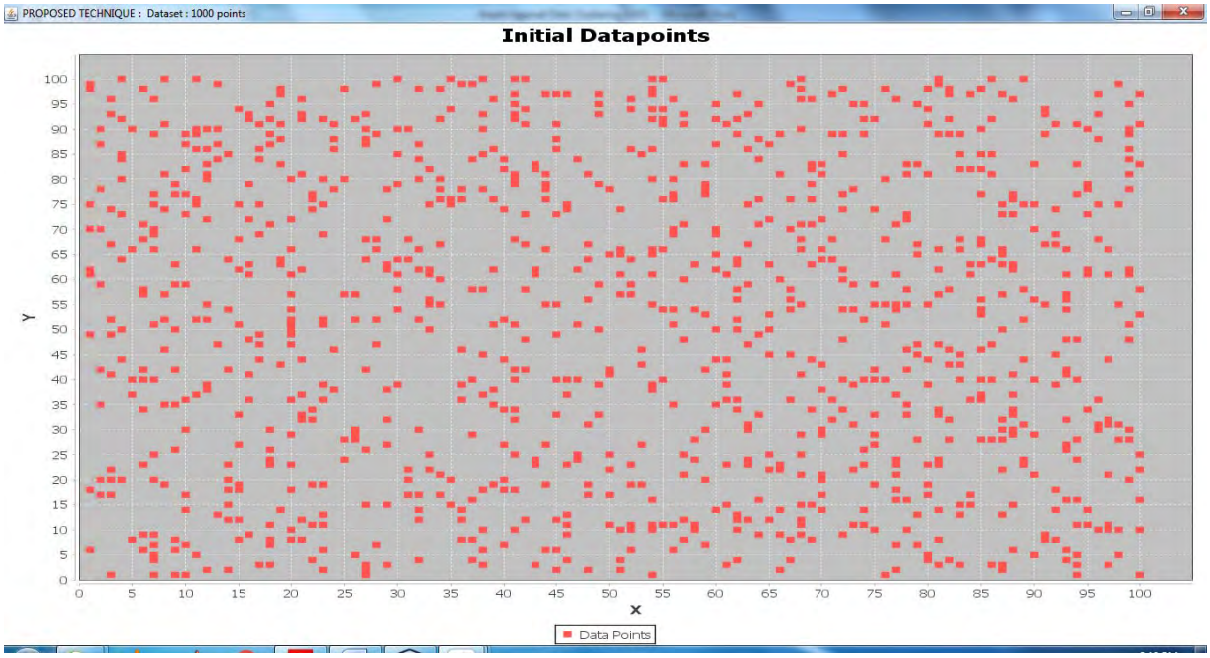


Fig. 4 Dataset selection.

7.2.3 Selection of first medoid

As shown in Fig. 5, the first initial medoid has been identified. The system takes the dataset and sorts the data points according to the distance from the origin. It then calculates a factor (dp/k) which is (1000/4) 250 in this case. It then calculates the mean of the first 250 elements of the dataset. It then identifies that element out of these 250 elements which has the smallest distance from the mean. This point is taken as the medoid. In this case the point (25, 24) has been identified as the first medoid element. The selected point has been represented by a red square while the normal data points are represented by blue circles.

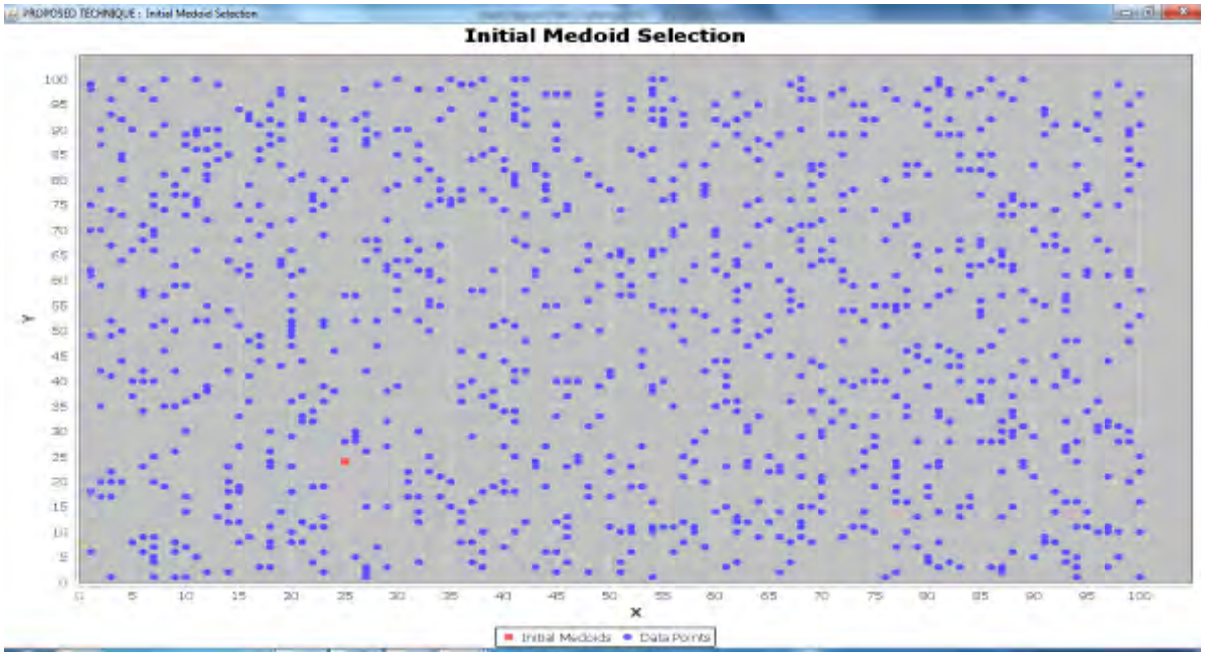


Fig. 5 Selection of first medoid.

7.2.4 Selection of second medoid

As shown in Fig. 6, the second initial medoid has been identified. The system calculates the mean of the next 250 elements of the dataset. It then identifies that element out of these 250 elements which has the smallest distance from the mean. This point is taken as the medoid. In this case the point (45,49) has been identified as the second medoid element. The already selected points have been represented by red squares while the normal data points are represented by blue circles.

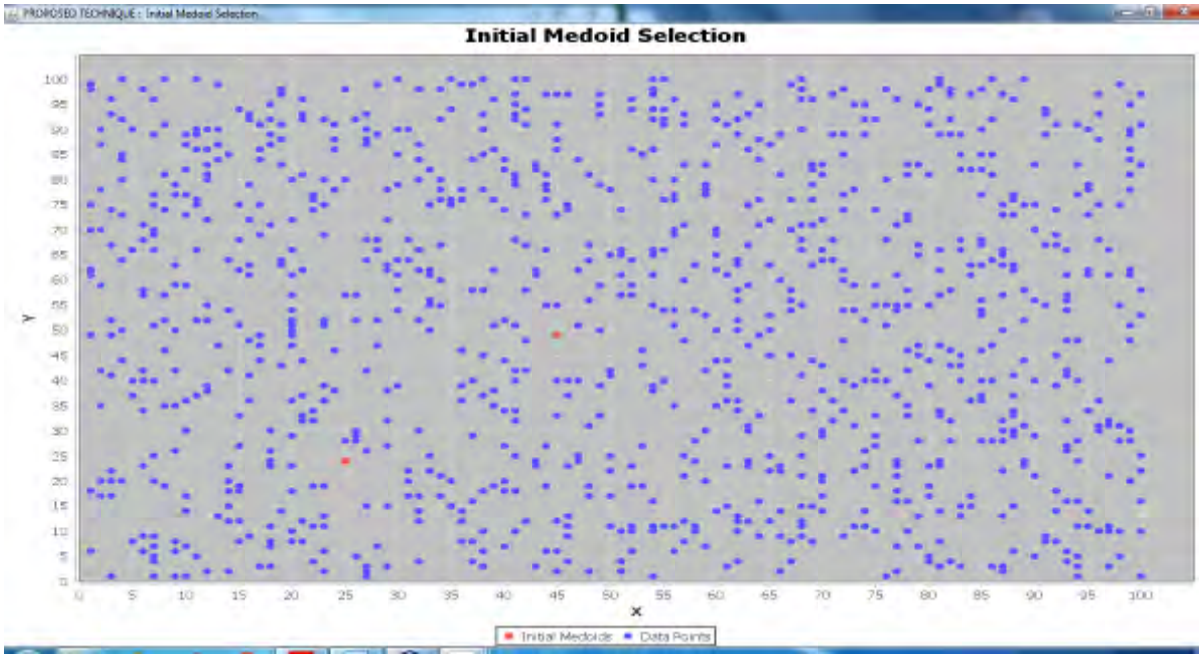


Fig. 6 Selection of second medoid.

7.2.5 Selection of third medoid

As shown in Fig. 7, the third initial medoid has been identified. The system calculates the mean of the next 250 elements of the dataset. It then identifies that element out of these 250 elements which has the smallest distance from the mean. This point is taken as the medoid. In this case the point (61,61) has been identified as the third medoid element. The already selected points have been represented by red squares while the normal data points are represented by blue circles.

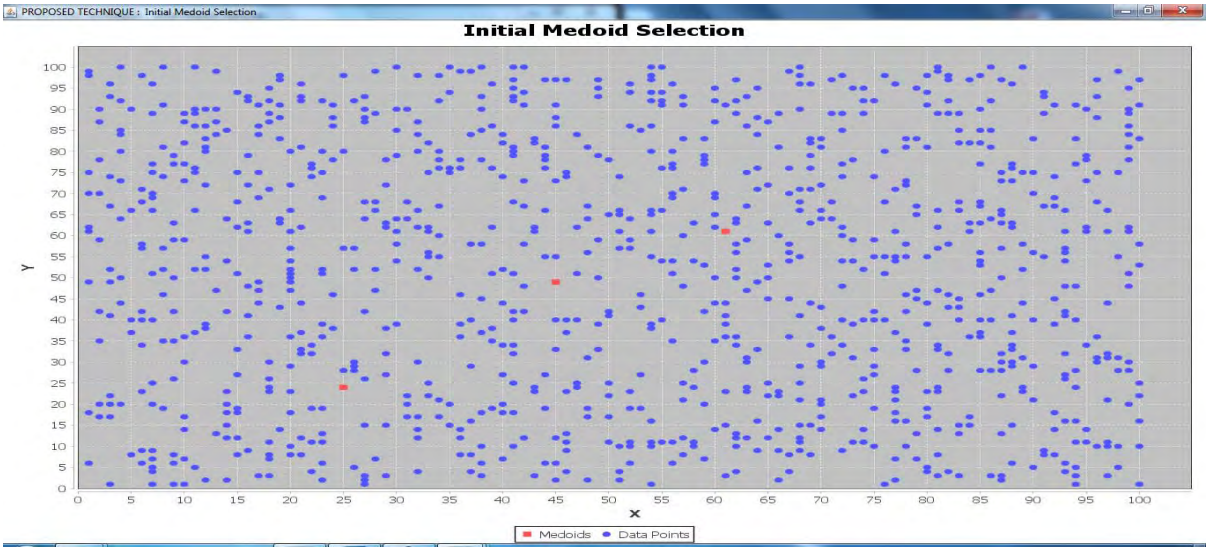


Fig. 7 Selection of third medoid.

7.2.6 Selection of fourth medoid

As shown in Fig. 8, the fourth initial medoid has been identified. The system calculates the mean of the next 250 elements of the dataset. It then identifies that element out of these 250 elements which has the smallest distance from the mean. This point is taken as the medoid. In this case the point (73,78) has been identified as the fourth medoid element. The already selected points have been represented by red squares while the normal data points are represented by blue circles.

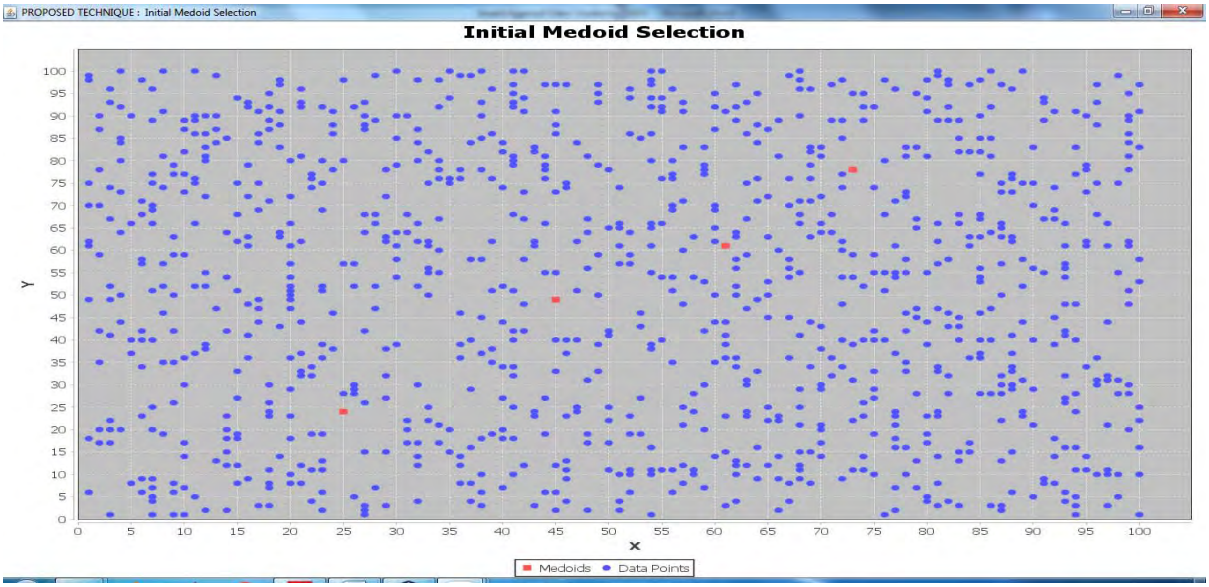


Fig. 8 Selection of fourth medoid.

7.2.7 Clustering of data

As shown in Fig. 9, four clusters are obtained from the dataset based on the initial medoids already selected. The system takes each data point and computes the distance between the data point and the already selected medoids. It then associates each data point with the medoid with which it has the smallest distance and adds the data point to the cluster represented by that medoid. In this, the elements of the first cluster are represented with yellow diamonds. The elements of the second cluster are represented with green triangles. The elements of the third and fourth cluster are represented by blue circles and red squares respectively.

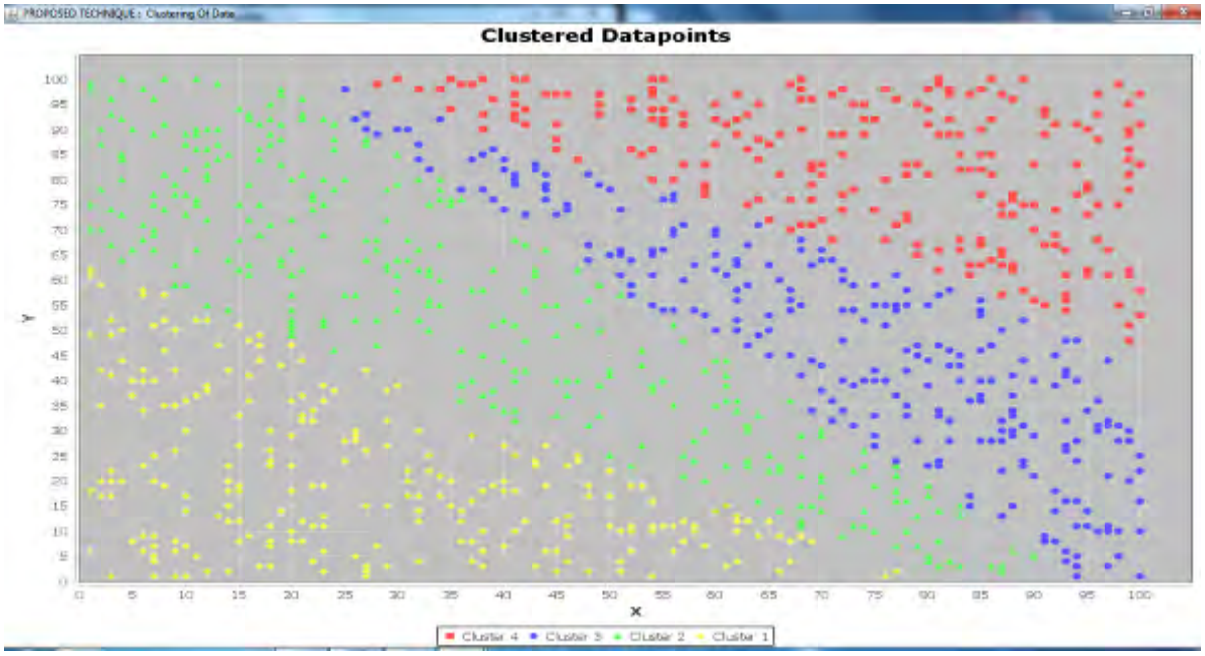


Fig. 9 Clustering of data.

7.2.8 Updation of medoids

As shown in Fig. 10, four new medoids have been identified based on the clustering obtained in the previous stage. The system moves from the initial medoids to other data points in the same cluster such that the cost of the cluster configuration is minimized. This way a new element or medoid is chosen to represent each data cluster such that the cost of the cluster configuration is minimum. The point (23,19) is chosen as the new medoid for cluster 1. The point (33,61) is chosen as the new medoid for cluster 2. The point (72,48) is chosen as the new medoid for cluster 3. The point (78,83) is chosen as the new medoid for cluster 4. Elements of the first cluster are represented with pink rectangles. Elements of the second, third and fourth cluster are represented by yellow diamonds, green triangles and blue circles respectively. The new medoids are represented with red squares.

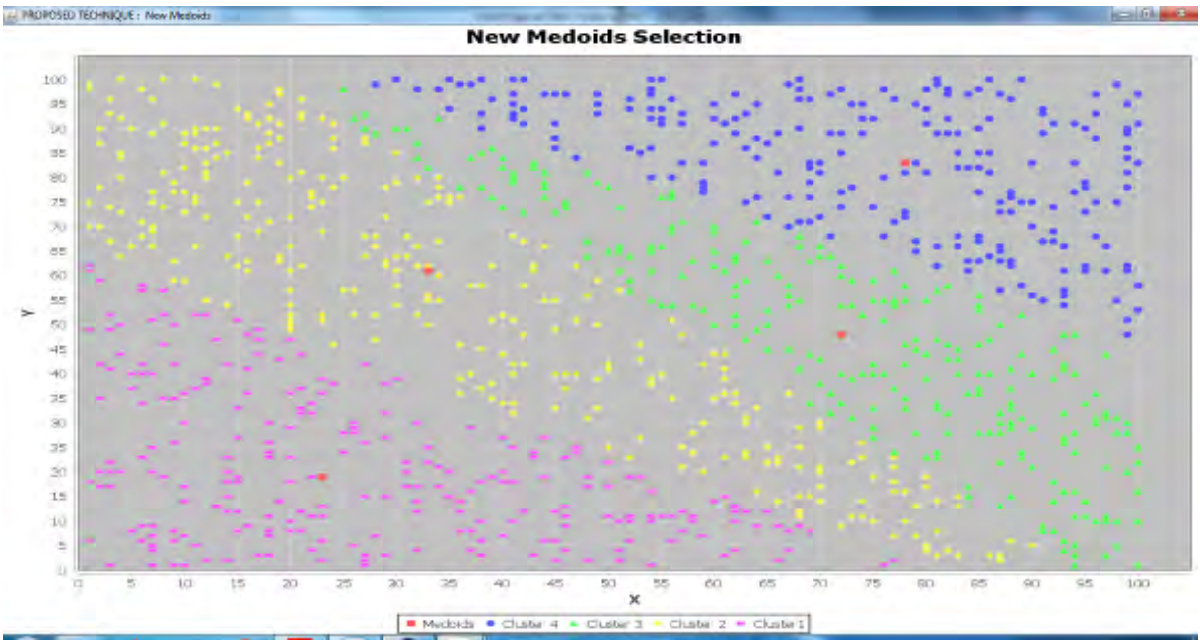


Fig. 10 Updation of medoids.

7.2.9 Outlier detection

As shown in Fig. 11, outlier data points have been identified for each cluster. The system takes the new set of medoids and calculates the average distance between the medoid and the data elements in the cluster. A threshold is calculated by using a multiplying factor of 1.5 with the average distance obtained for each cluster. All elements which lie at a distance greater than the threshold from their medoids are identified as outliers. All elements which lie at a distance less than or equal to the threshold are termed as inliers for that cluster. The inlier and outlier elements of cluster 1 are represented with blue circles and green triangles respectively. The inlier and outlier elements of cluster 2 are represented with yellow diamonds and pink rectangles respectively. The inlier and outlier elements of cluster 3 are represented with maroon rectangles and grey semi inverted triangles respectively. The inlier and outlier elements of cluster 4 are represented with light orange rectangles and light blue inverted triangles respectively.

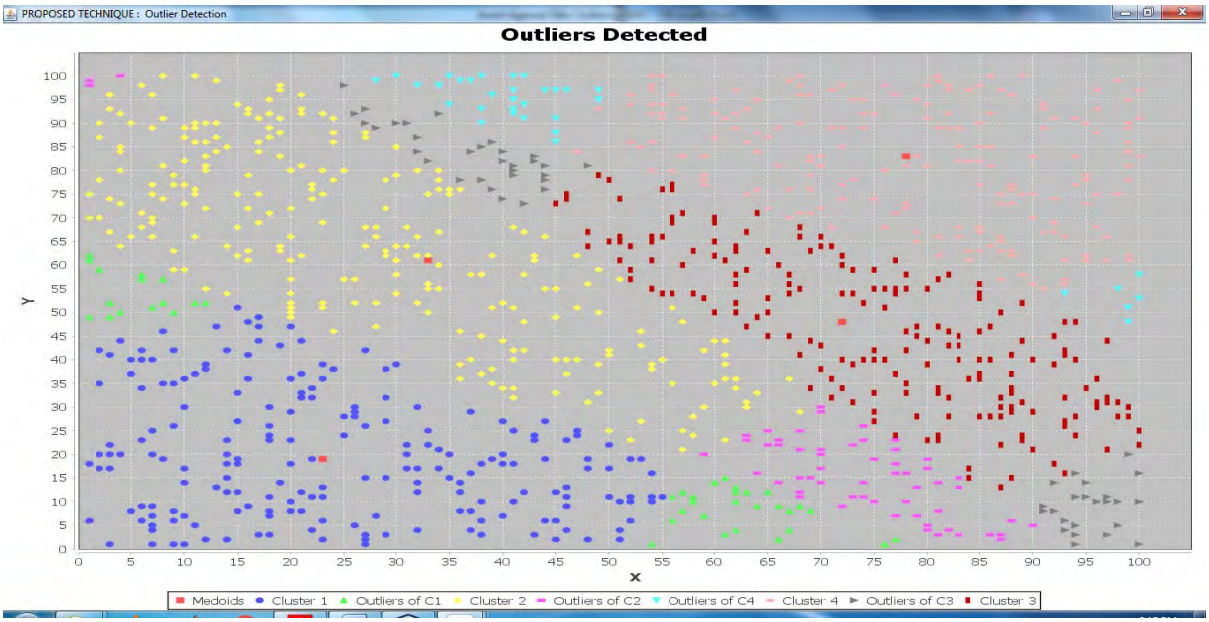


Fig. 11 Outlier detection.

7.2.10 Results

A shown in Fig. 12, the system provides the results that are obtained during the execution of the system. The system provides a breakdown about the number of inlier and outlier elements present in each cluster. In this case it can be seen that the system got 197 inlier and 41 outlier elements for the first cluster. In the case of the second cluster, 240 inlier and 58 outlier elements were detected. For the third cluster, 189 inlier and 54 outlier elements have been detected. Similarly for the fourth cluster, 187 inlier points and 34 outlier points have been detected. At the end, the total number of outliers detected and the execution time is provided by the system. In this case, the system detects a total of 187 outliers with an execution time of 1030 ms.

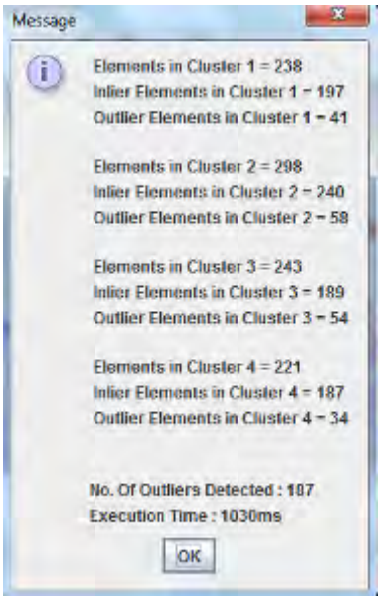


Fig. 12 Results.

8 Analysis and Results

The performance of the clustering algorithms is presented in this section. In Section 8.1, the performance of the proposed work is computed in terms of the number of outliers detected by the technique. In Section 8.2, the proposed work is compared against the already existing PAM algorithm in terms of outliers detected. In Section 8.3, the performance of the proposed work is computed in terms of the time taken for execution. In Section 8.4, the proposed work is compared against the already existing PAM algorithm in terms of the time taken for execution. Ten different datasets comprising of different number of data points have been chosen for the analysis of the proposed work. The number of clusters is taken as 4. The PAM algorithm is run 5 times and the average values of these runs are taken for comparison.

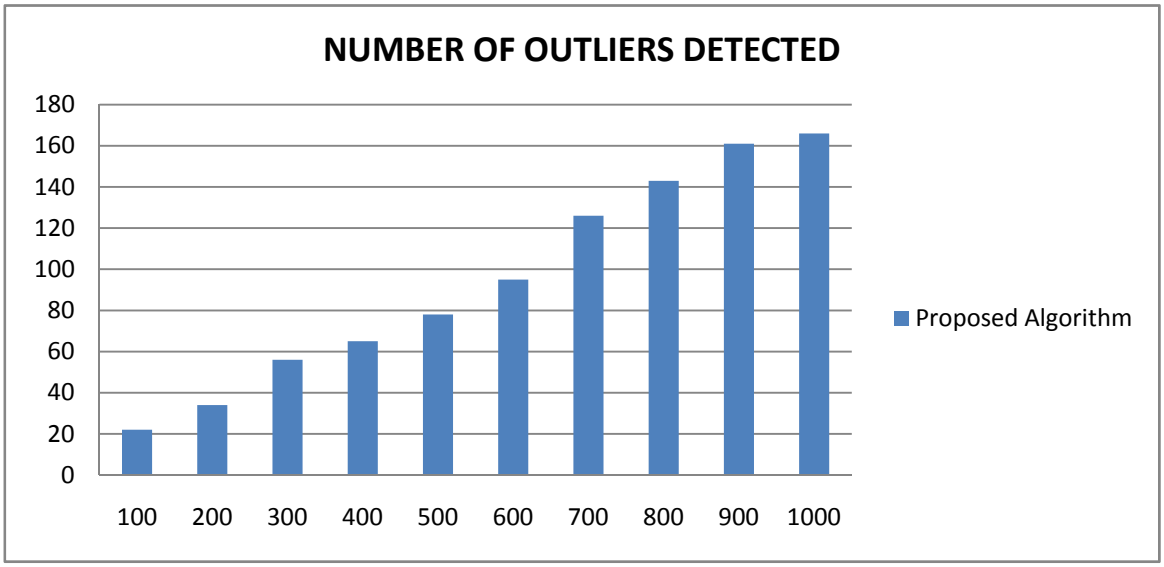
8.1 Outlier accuracy for proposed technique

In this section, the actual number of outliers detected by the proposed algorithm is calculated. Datasets comprising of different number of data points are considered and the results obtained are presented in Table 1.

The number of outliers detected by the proposed algorithm when applied to different data sets is mentioned in Table 1. It can be seen that the proposed technique detects 22 outliers when applied to 100 data points. Similarly, the proposed technique detected 166 outliers when it was applied to 1000 data points.

Table 1 Number of outliers detected by proposed algorithm.

Number Of Data Points	Outliers Detected
100	22
200	34
300	56
400	65
500	78
600	95
700	126
800	143
900	161
1000	166



Graph 1 Number of outliers detected by proposed algorithm.

Graph 1 shows a graphical representation of the outliers detected by the proposed algorithm when applied to different number of data points. As can be seen from the graph the proposed algorithm detected 22 outliers when applied on the dataset containing 100 data points while 166 outliers were detected when it was applied on the dataset consisting of 1000 data points.

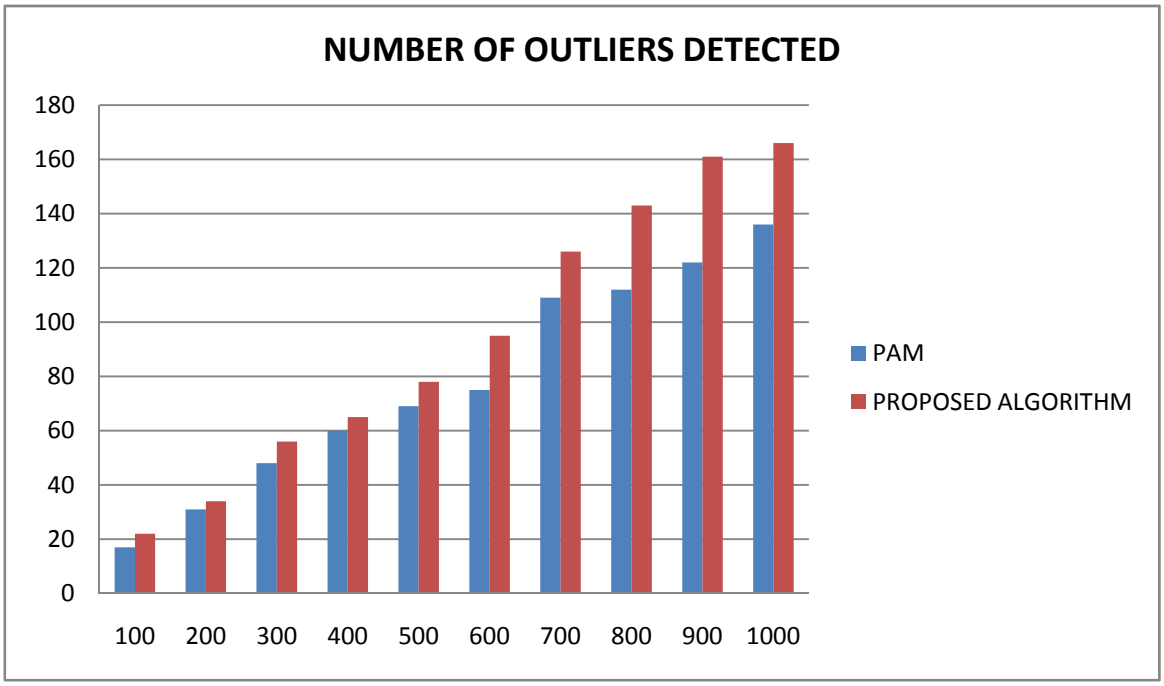
8.2 Comparison of outliers detected between PAM and proposed algorithm

In this section, the actual number of outliers detected by the proposed algorithm and the already existing PAM algorithm is compared. Datasets comprising of different number of data points are considered and the results obtained are presented in Table 2.

The number of outliers detected by PAM and the proposed algorithm when applied to different data sets is mentioned in Table 2. It can be seen that the proposed technique detects 22 outliers when applied to 100 data points while the PAM technique detects 17 outliers. Similarly, the proposed technique detected 166 outliers when it was applied to 1000 data points whereas PAM detected 136 outliers. It can be seen that the proposed technique detects more number of outliers than PAM in every case and thus the proposed technique is better than PAM.

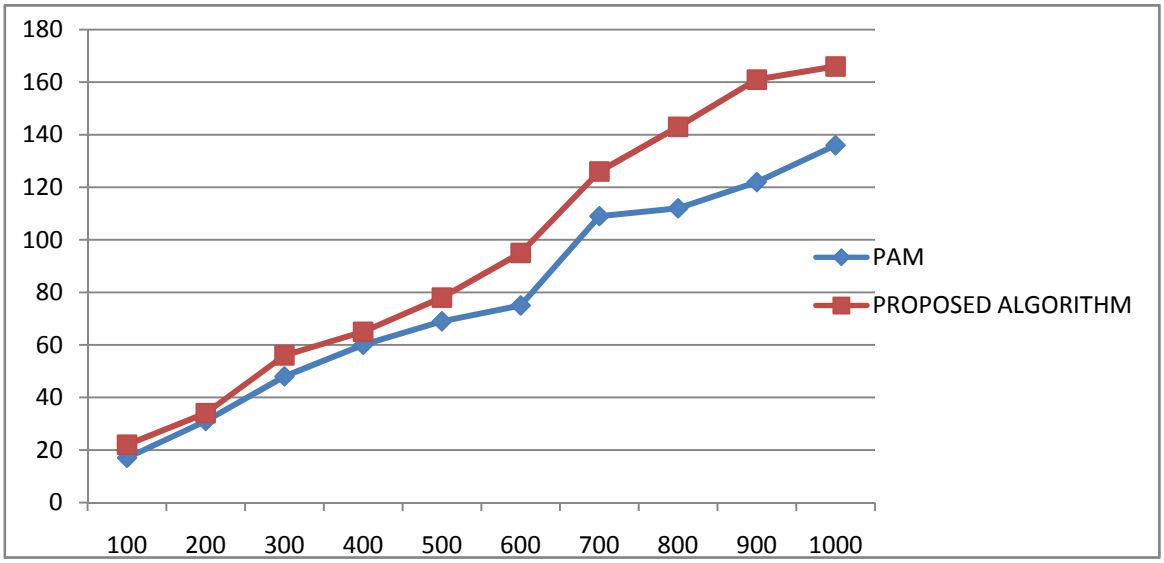
Table 2 Outliers detected comparison between PAM and proposed technique.

Number Of Data Points	Outliers Detected by PAM	Outliers Detected by Proposed Technique
100	17	22
200	31	34
300	48	56
400	60	65
500	69	78
600	75	95
700	109	126
800	112	143
900	122	161
1000	136	166



Graph 2 Outliers detected comparison between PAM and proposed technique.

Graph 2 shows a graphical representation of the outliers detected by PAM and the proposed algorithm when applied to different number of data points. As can be seen from the graph the proposed algorithm detected 22 outliers when applied on the dataset containing 100 data points while 17 outliers were detected by PAM. Similarly, the proposed algorithm detected 166 outliers when applied on the dataset consisting of 1000 data points while PAM detected 136 outliers. By observing the results it can be said that the proposed algorithm is better than the already existing PAM.



Graph 3 Outliers detected comparison between PAM and proposed technique

Graph 3 shows a graphical representation about how the number of outliers detected by each technique increases when the number of data points is increased. It can be seen in Graph 3 that the proposed algorithm detects more number of outliers than PAM in all the cases and hence is the better technique.

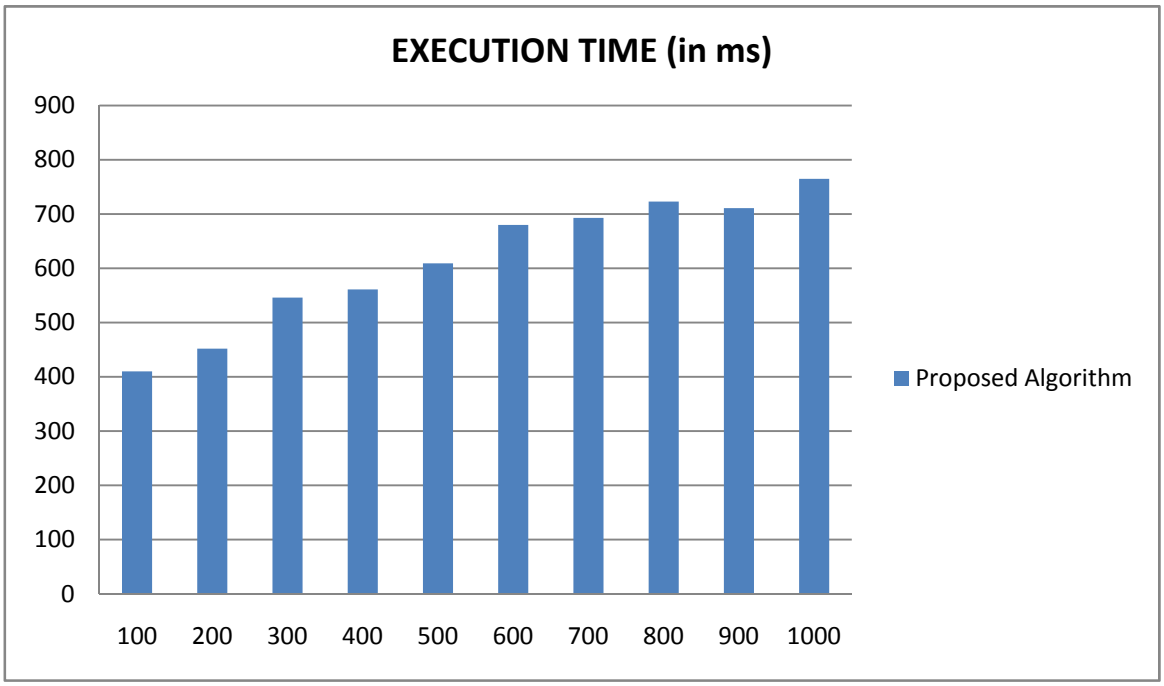
8.3 Time complexity of proposed technique

In this section, the actual execution time of the proposed algorithm is calculated. Datasets comprising of different number of data points are considered and the results obtained are presented in Table 3.

Table 3 Execution time of proposed algorithm.

Number Of Data Points	Execution Time(in ms)
100	410
200	452
300	546
400	561
500	609
600	680
700	693
800	723
900	711
1000	765

The execution time of the proposed algorithm when applied to different data sets is mentioned in Table 3. It can be seen that the proposed technique took 410 ms to execute when applied to 100 data points. Similarly, the proposed technique took 765 ms to execute when it was applied to 1000 data points.



Graph 4 Execution time of proposed algorithm.

Graph 4 shows a graphical representation of the execution time of the proposed algorithm when applied to different number of data points. As can be seen from the graph the proposed algorithm took 410 ms to execute when applied on the dataset containing 100 data points while 765 ms were required by the algorithm to execute on the dataset consisting of 1000 data points.

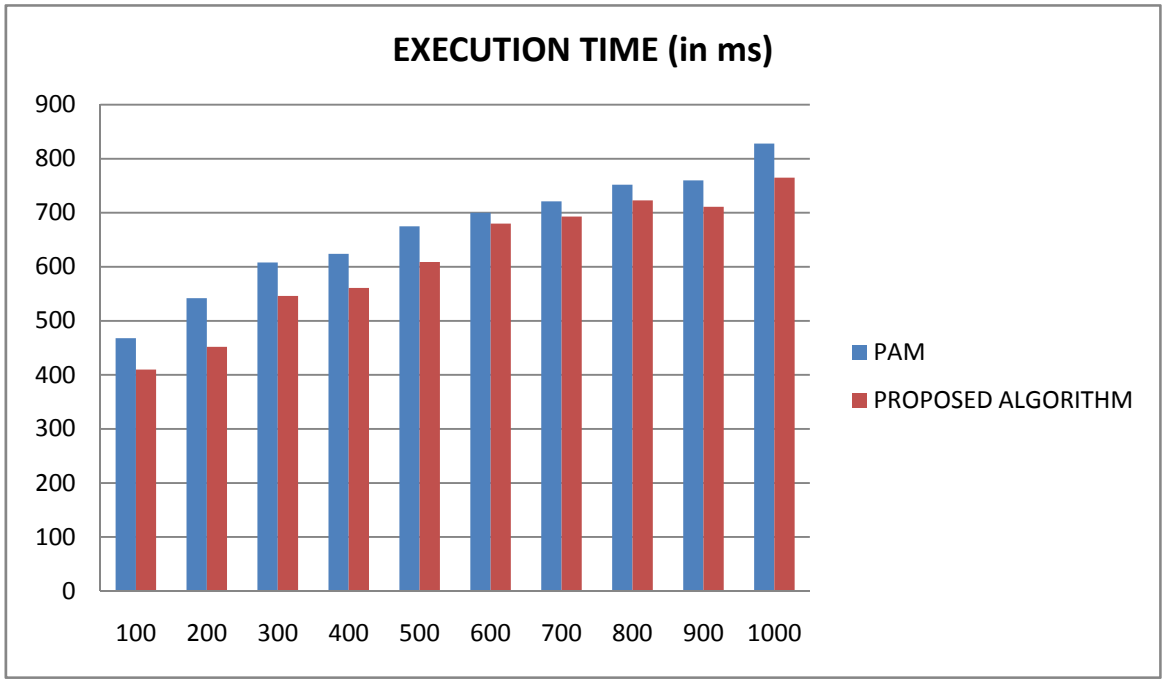
8.4 Comparison of time complexity between PAM and proposed algorithm

In this section, the actual execution time of the proposed algorithm and the already existing PAM algorithm is compared. Datasets comprising of different number of data points are considered and the results obtained are presented in Table 4.

Table 4 Time complexity comparison between PAM and proposed technique.

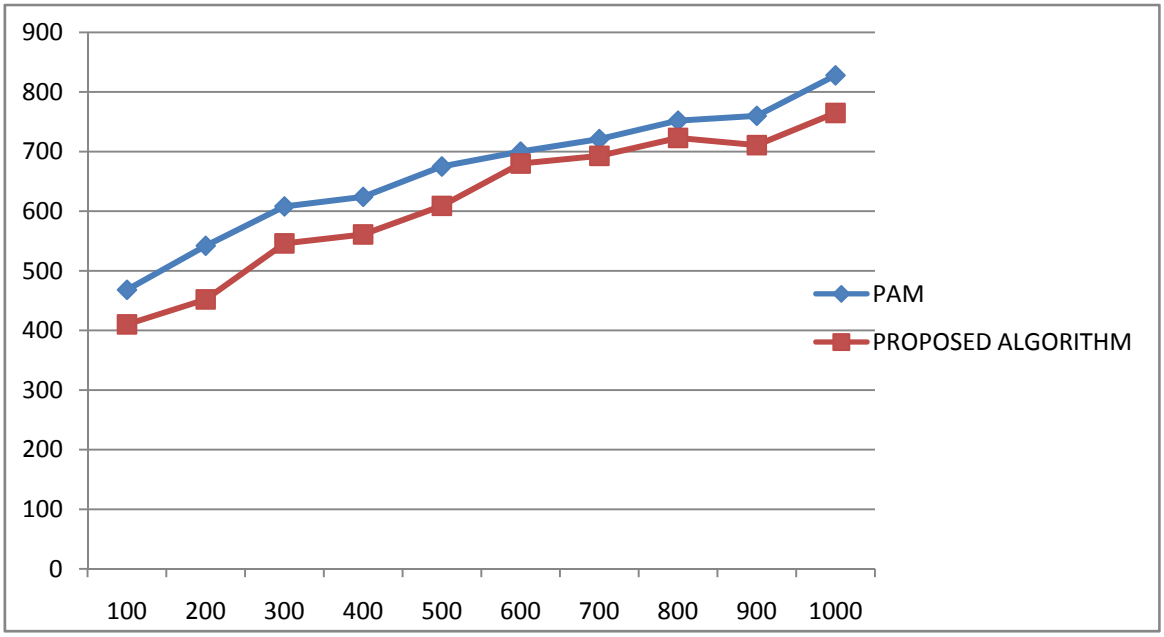
Number Of Data Points	Execution Time(in ms) of PAM	Execution Time(in ms) of Proposed Technique
100	468	410
200	542	452
300	608	546
400	624	561
500	675	609
600	700	680
700	721	693
800	752	723
900	760	711
1000	828	765

The execution time of PAM and the proposed algorithm when applied to different data sets is mentioned in Table 4. It can be seen that the proposed technique takes 410 ms to execute when applied to 100 data points while the PAM technique took 468 ms to execute. Similarly, the proposed technique took 765 ms to execute when it was applied to 1000 data points whereas PAM took 828 ms to execute. It can be seen that the proposed technique is faster than PAM in every case and thus the proposed technique is better than PAM.



Graph 5 Time complexity comparison between PAM and proposed technique.

Graph 5 shows a graphical representation of the execution time of PAM and the proposed algorithm when applied to different number of data points. As can be seen from the graph the proposed algorithm took 410 ms to execute on the dataset containing 100 data points while 468 ms were required by PAM to execute. Similarly, the proposed algorithm took 765 ms to execute when applied on the dataset consisting of 1000 data points while PAM took 828 ms to execute. By observing the results it can be said that the proposed algorithm is better than the already existing PAM.



Graph 6 Time complexity comparison between PAM and proposed technique

Graph 6 shows a graphical representation about how the execution time of each technique increases when the number of data points is increased. It can be seen in Graph 6 that the proposed algorithm is faster than PAM in all the cases and hence is the better technique.

9 Conclusion

Data mining is the process of extracting data from the data set. In data mining, clustering is the process of grouping data that have high similarity in comparison to one another. There are different algorithms that exist for detecting outliers. In this paper, a new clustering technique has been proposed for outlier detection. The goal of the algorithm presented in the paper is to improve the process of outlier detection. The experimental results show that the proposed algorithm improves the time taken and the outliers detected over the already existing technique. Hence it can be said that the proposed technique is better than the existing technique.

References

Aggrwal S, Kaur P. 2013. Survey of partition based clustering algorithm used for outlier detection. International Journal For Advance Research In Engineering and Technology, 1(5): 57-62

Al-Zoubi M. 2009. An effective clustering-based approach for outlier detection. European Journal of Scientific Research, 28(2): 310-316

Bhawna N, Ahirwal P, Salve S, Vamney S. 2011. Document Classification using Expectation Maximization with Semi Supervised Learning. International Journal on Soft Computing, 2(4): 37-44

Bo F, Wenning H, Gang C, Dawei J, Shuining Z. 2012. An Improved PAM Algorithm for Optimizing Initial Cluster Center. IEEE International Conference on Computer Science and Automation Engineering, 24-27

Duraj A, Zakrzewska D. 2014. Effective Outlier Detection Technique with Adaptive Choice of Input Parameters. Proceddings of the 7th IEEE International Conference on Intelligent Systems, 322: 535-546

Jain N, Srivastava V. 2013. Data Mining Techniques: A Survey Paper. International Journal Of Research in

Engineering and Technology, 2(11): 116-119

Loureiro A, Torgo L, Soares C. 2004. Outlier detection using clustering methods: a data cleaning application. In: Proceedings of KDNet Symposium on Knowledge-Based Systems for the Public Sector. Bonn, Germany

Karmaker A, Rahman SM. 2009. Outlier Detection in Spatial databases using Clustering Data Mining. Sixth International Conference on Information Technology: New Generations, 1657-1658

Kaur N, Mann AK. 2013. Survey Paper on Clustering Techniques. International Journal of Science, Engineering and Technology Research, 2(4): 803-806

Mansur MO, Noor M, Sap M. 2005. Outlier Detection Technique in Data Mining: A Research Perspective. In: Proceedings of the Postgraduate Annual Research Seminar. 23-31

Padhy N, Mishra P, Panigrahi R. 2012. The Survey Of Data Mining Applications And Feature Scope. International Journal Of Computer Science, 2(3): 43-58

Paterlini AA, Nascimento MA, Junior CT. 2011. Using Pivots to Speed Up k-Medoids Clustering. Journal of Information and Data Management, 2(2): 221-236

Rajagopal S. 2011. Customer Data Clustering Using Data Mining Techniques. International Journal of Database Management Systems, 3(4): 1-11

Singh K, Upadhyaya S. 2012. Outlier Detection: Applications and Techniques. International Journal of Computer Science, 9(1): 307-323

Smita, Sharma P. 2014. Use of Data Mining in Various Field: A Survey Paper. IOSR Journal of Computer Engineering, 16(3): 18-21

Vaghela D, Maitry N. 2014. Survey on Different Density Based Algorithms on Spatial Dataset. International Journal of Advance Research in Computer Science and Management Studies, 2(2): 362-366

Vijayarani S, Nithya S. 2011. An efficient clustering algorithm for outlier detection. International Journal of Computer Applications, 32(7): 22-27

Yadav J, Sharma M. 2013. A Review of K-mean Algorithm. International Journal of Engineering Trends and Technology, 4(7): 2972-2976

Zhang L, Yang M, Lei D. 2012. An improved PAM clustering algorithm based on initial clustering centers. Applied Mechanics and Materials, 135-136: 244-249

Zhang WJ. 2016. A method for identifying hierarchical sub-networks / modules and weighting network links based on their similarity in sub-network / module affiliation. Network Pharmacology, 1(2): 54-65

Zhang WJ, Li X. 2016. A cluster method for finding node sets / sub-networks based on between- node similarity in sets of adjacency nodes: with application in finding sub-networks in tumor pathways. Proceedings of the International Academy of Ecology and Environmental Sciences, 6(1): 13-23

Zhang WJ, Qi YH, Zhang ZG. 2014. Two-dimensional ordered cluster analysis of component groups in self-organization. Selforganizology, 1(2): 62-77