IMPROVISATION OF DBSCAN CLUSTERING ALGORITHM USING VARIOUS METRICS

[1]Valarmathy.N [2] and Dr.Krishnaveni Sakkarapani

[1]Research Scholar & Assistant Professor, PG & Research Department of Computer Science, Pioneer College of Arts & Science, Coimbatore, Tamil Nadu, India
valarmathykamalam@gmail.com

[2]Assistant Professor, Department of Computer Science, PSGR Krishnammal College for Women, Coimbatore, Tamil Nadu, India
sss.veni@gmail.com

ABSTRACT
The primary goal of data mining is to organize data into meaningful clusters using several clustering algorithms. The DBSCAN algorithm, which may be used for a variety of applications, is the most effective of the numerous clustering techniques. This technique is a high-quality density-based method with various advantages, including the ability to identify arbitrarily shaped clusters, and the ability to identify outliers and ignore them before clustering. Epsilon (Eps) and a minimal number of points (MinPts) are the two key input factors that have a significant impact on clustering performance. To overcome this problem, the K-distance graph approach is used to automatically choose Eps and MinPts, and spatial access methods are used to speed up neighbourhood calculation for each data point. The suggested new technique makes use of the spatial access method and the k-distance graph method to improve scalability and accelerate the execution process. The results of the experiments clearly show that combining the K-Distance tree method with the DBSCAN algorithm is efficient in terms of all metrics and can cluster both high and low dimensional data effectively.
Keywords: *DBSCAN, Outlier detection, Speed Optimization, Nearest Neighbour Search*

1. INTRODUCTION

Clustering and classification are two fundamental data mining techniques for evaluating patterns and structure in a database [29]. The clustering technique separates data objects depending on the degree of similarity between two mutual data elements. This clustering partition is accomplished by splitting a group of items into subsets of similar objects, which are referred to as clusters; however, the objects within each cluster are similar to those within it and distinct from that outside. Hierarchical based algorithms, spectral algorithms [24], grid-based algorithms [23], density-based algorithms [33], and partition-based algorithms [13] are the four primary kinds of clustering algorithms. These methods differ in characteristics such as (i) the algorithm or procedures utilized to compute the similarity index (within and between clusters) and (ii) finding the appropriate threshold for recognizing and grouping cluster elements. (iii) The technique used to categorize the items into one or more clusters of varying degrees [34]. As a result, the aggregated structure is used, and the objects may be successfully retrieved. Many applications, such as image segmentation, web data mining, and information retrieval, utilize this clustering technique.

Data mining requires the partitioning of data into meaningful clusters. Because the success rate is determined by the algorithm used, Density-Based Spatial Grouping of Applications with Noise (DBSCAN) has emerged as a popular method for clustering low and high-dimensional data. The problem of calculating the neighbourhood for each data object is a time-consuming operation that is addressed in this approach by applying spatial access methods [14]. DBSCAN's performance for clustering high-dimensional data has to be enhanced so that large datasets can be clustered in a short amount of time. The downside of this algorithm is its time complexity and execution time. Thus to overcome this issue a modern cross breed calculation utilizing a K-Distance graph is utilized together with the DBSCAN calculation [20]. Hence the programmed determination of Eps and Minpts will increment the speed of the clustering handle. The looking time required to create clusters can be

decreased by utilizing the KD-Tree calculation. Hence our proposed unused calculation can decrease the neighbourhood look time so that the general clustering time is decreased.

2. LITERATUREREVIEW

The time complexity issue seen in the DBSCAN algorithm is solved by many researchers and it is broadly classified as a sampling and space partitioning method which can minimize the number of core points (pruning).The variation of DBSCAN includes SDBSCAN and Rough-DBSCAN. In all these techniques, a trade-off between sampling rates and speed of execution exists. That is, low sampling rates reduce the execution time requirement but will degrade the clustering performance in terms of accuracy[1].

FDBSCAN and IDBSCAN have been used as a clever middle detection approach which reduces the processing time [2]. Generally, the sampling approach is mixed with the middle factor detection approach to enhance the processing time [4].SPARROW [35] approach is used to lessen the number of middle factors. The border factors have been frequently recognized as outliers or noise.

Many parallel methods have been implemented along with DBSCAN and are presented in [7] which reduces the execution time but the final output obtained is not satisfactory. In the partition-based method, each partition is recognized at a pre-processing stage but it involves a tedious process for determining the correct parameters for input [21].

I-DBSCAN is a new algorithm which makes use of two kinds of prototypes like coarser level and finer level. The processing time is reduced in coarser level and finer level helps to increase the deviated result [31].

MEDBSCAN [15] makes use of a novel parameter known as MDV (Maximum Distance Value) along with the query region to identify those objects which fall in between the MDV value and the range of epsilon neighbourhood. This parameter reduces the number of region queries and the processing time.

Many space partitioning algorithms have been proposed like CLARINS [32], K-Means simple one-pass clustering [16],and Space indexing technique like R-tree or X-tree is used for retrieving the objects neighbours faster than normal methods.

FDBSCAN [35] algorithm is a combination of two different algorithms for the selection of representative objects from the neighbourhood. A MANET network faces a lot of issues and challenges. To ensure the correct transmission and delivery of the data packets the network is divided into number of clusters. Density based clustering algorithm is used. Security is another great challenge. To ensure that the transmitted data is secure, a cryptographic technique is incorporated into the network [25-28,30].

Hencil JPeter and A, Antonysamy [12] proposed a new ODBSCAN algorithm which is a combination of FDBSCAN and MEDBSCAN. It works by selecting only four representative points for the expansion of clusters.

3. DBSCAN: AN OVERVIEW

In this algorithm, the dense regions are identified by measuring the data objects that are very close to the given point. The points closer to each other are grouped using distance function. This algorithm can also identify the outliers or noise, the points in the low-density regions are identified as noise [3], Patterns are identified and the relationships between data objects are also predicted.

ParameterEstimation

The success of an algorithm depends on the right parameters and the values assigned to them.This DBSCAN algorithm require two main parameters such as Eps and MinPts.

Eps: The parameter Eps(epsilon) denotes the radius of the neighbourhood around the point The Eps value to be chosen must be based on the distance of the data object (here K-Distance tree method) is used to find out the Eps value. It should neither be too small nor too high

MinPoints: The minimum number of points needed to form a dense region is denoted by MinPoints. To set the MinPts parameter as 5 at least 5 points are needed to form a dense region. The assumed MinPts based on the number of dimensions (D) must be greater than D+1 [Minpts> D+1]. By default, the minimum value assigned to the Minpts is 3, but according to the size of the dataset the values should be chosen. It denotes the number of neighbours within the Eps radius [10].

For example: By assigning any point x in the dataset, the core points are marked as points that are greater than or equal to MinPts and Border points are those which are less than Minpts and which are equal to X are those core points Z are epsilon neighborhood of x. if the point is neither in core nor a border point then it is treated as a noise point or an outlier [20].

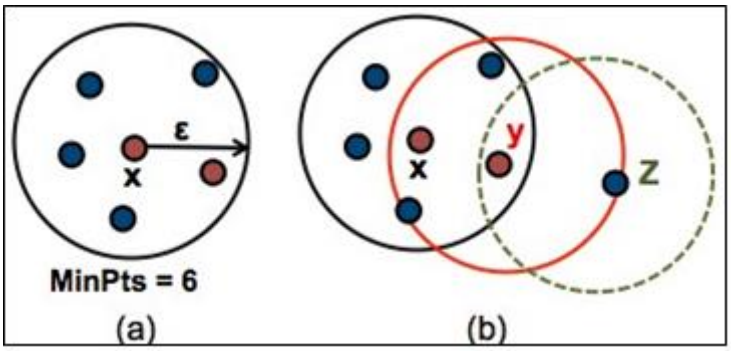

Figure 1: a.Clustering Point b.DBSCAN Clustering Process

The figure 1 illustrates the different types of clustered points (core, border and outlier) available using **MinPts = 6**.

Here x is a core (centre) point because neighbours_epsilon(x) = 6, y is a border point because neighbours_epsilon(y) <MinPts, but it belongs to the epsilon-neighbourhood of the core point x.Atlast the outlier point is z.

The three main terminologies commonly used in DBSCAN algorithm are:

A group of density connected points is known as density-based cluster. The process of identifying the density connected points is performed by using the algorithm is shown in figure 2.

- **Direct density reachable**: A point "A" is **directly density reachable** from another point "B" if:
  i) "A" is in the epsilon-neighbourhood of "B" and
  ii) "B" is a core point
- **Density reachable**: A point "A" is **density reachable** from "B" if there are a set of core points leading from "B" to "A
- **Density connected**: Two points "A" and "B" are **density connected** if there exist a core point "C", such that both "A" and "B" are **density reachable** from "C"

- **For every point Xi, calculate the distance between Xi and the other points. Identify all neighbour points that lies within the distance Eps of the starting point Xi. Then those points, with a neighbour_count >= MinPts, is marked as core point or visited.**
- **For every core point, if the cluster is not assigned then create a new cluster. Then findout iteratively all the density connected points and group them to the same cluster as the core point.**
- **Repeat the process through all the remaining unvisited points in the dataset.**

The points which does not belong to any cluster are then spotted as outliers or noise and finally removed.

Figure 2: Algorithm of Density-based Clustering

Pseudocode of the DBSCAN algorithm is given in figure 3.

```
DBSCAN(DB, distFunc, eps, minPts)
 {
        C = 0                                   /* Intialize Cluster counter */
 for each point P in database DB
 {
if label(P) ≠ undefined then continue           /* inner loop is processed already */
        Neighbours N = RangeQuery(DB, distFunc, P, eps)    /* neighbours are found */
if |N| < minPts then {                          /* check the density value */
        label(P) = Noise                        /* Noise point is labelled */
continue
 }
        C = C + 1                               /* fetch next cluster label */
        label(P) = C                            /* initial point is labelled */
        Seed set S = N \ {P}                    /* expansion to identify neighbours */
for each point Q in S {                         /* every seed point is being processed */
if label(Q) = Noise then label(Q) = C           /* Noise to border point is changed */
if label(Q) ≠ undefined then continue           /* processed previously */
        label(Q) = C                            /* neighbour are labelled */
Neighbors N = RangeQuery(DB, distFunc, Q, eps)          /* neighbours are found */
if |N| ≥ minPts then {                          /* check the density value again */
        S = S ∪N                                /* new neighbours are added to seed set */
        }       }   }       }
```

Figure 3: Pseudocode of traditional DBSCAN algorithm

Merits of DBSCAN

- No prior knowledge of the number of cluster is required.
- This algorithm can find out clusters on arbitrary shape, even if it not completely surrounded by a different cluster. The problem of single link effect is performed using Minpts parameter, which is assigned and being connected to different clusters through at h in line of points [19].
- The Identification of outliers improves its Performance.
- DBSCAN algorithm is not sensitive in ordering the data points in the database it requires only two main parameters like Eps and Minpts only [21].
- DBSCAN is intended for use with huge databases that can speed up region queries using an R* tree [9].
- The parameters Minpts and ε(Eps) can be set only by a domain expert, who has understood the data very clearly.

Demerits of DBSCAN

- DBSCAN is not completely deterministic [8].
- The distance measure dist (ε) used in DBSCAN helps to achieve good quality of result. The default distance metric used is Euclidean distance, but for high dimensional data this function is almost useless and this problem is called as "Curse of Dimensionality" hence it is very difficult to identify the value of Eps in case of high dimensional data [5].
- When the difference in the densities is larger than DBSCAN cannot cluster datasets and choosing MinPts and Eps cannot be done appropriately [11].
- Choosing a meaningful distance threshold ε is very difficult if the data and scale are not clearly understood [18].

4. PROPOSED METHODOLOGY

K-Distance Tree based DBSCAN (KDTDBSCAN)

The above mentioned two problems can be solved by identifying Epsilon and Minpts value automatically using KD-tree algorithm. The search complexity is also reduced in KD-tree algorithm. KD-Tree method performs segmentation of data structure by arranging the points in k- dimensional matrix space. KD tree can be used in situation of having to use multi-dimensional key, if there are two nodes and k-dimensional point and it is called as binary tree. The non-leaf node is considered as a splitting hyperplane which can divide the search space into right and left subspaces [17]. The algorithm for balanced KD- tree for the list of n given points is given in figure 4.

```
Input  : List of points pointList and depth
Output  : KD Tree
Function kdtree(pointList, depth)
Step 1 : Select axis based on depth so that axis cycles through all valid values
(axis = depth mod k)
Step 2 : Sort point list and choose median as pivot element
Step 3 : Create node and construct subtrees
node location := median;
        leftChild := kdtree(points in pointList before median, depth+1);
rightChild := kdtree(points in pointList after median, depth+1);
Step 4 : Repeat Steps 1 - 3 till pointList in empty.
```

Figure 4:KD-TreeAlgorithm

1. *The process starts with the root node and moves downward in the tree by inserting the search point repeatedly.*
2. *Once when leaf node is reached, the current node is save as the "current best"*
3. *This procedure repeats the recursion process in the tree and the following process is performed in each node:*
   a) *If the current node value chosen is closer to the current best value, then it becomes the current best value.*
   b) *Both the sides of the splitting plane are checked to find out the point closer to the search point and assign it as the current best. The splitting of hyperplane is done e with a hypersphere around the search point which has the radius of current nearest distance.*
      - *If the hypersphere crosses the plane, there is a possibility of nearer points on the other side of the plane. So this algorithm moves down to other branch of the tree from the current node to search closer points, also adopting the same iterative process for the complete search.*
      - *If the hypersphere does not intersect the splitting plane, then the algorithm continues goes up to the tree and the whole branch on the other side of that node is rejected.*
4. *The search process is complete, when the algorithm completes this process for the root node.*

Figure 5: Search Algorithm using KD-Tree

Usually we use Nearest Neighbour (NN) algorithm to estimate the points nearer to the given point. The searching through nearest neighbour using KD-Tree is given in figure 5 which can search very large datasets. Epsilon and MinPts parameters can be automatically detected using K-distance graph. Variable d is assigned to the distance of a point P. The d-neighbourhood contains K+1 points, if there are many points that has exactly the same distance d from p. Changing the value of point k with in a cluster group will not change the value of d. This is because of K^{th} nearest neighbours of p for k. Generally K is allocated with 1, 2, 3 and so on, so that, it approximately lies on a straight line.

Step 1: Calculate the distance between each point to all other points from it and compute average to find the average distance using the formula

$$d(Pi) = \frac{\sum_{i=1}^{n} distance(Pi, Xi)}{2(n-1)}$$

Step 1.1: Then calculate avg(d), which is the average of all d(Pi).

$$avg(d) = \frac{\sum_{i=1}^{n} d(Pi)}{n}$$

Step 2: For every point Pi in the educational dataset, a circle is drawn with Pi as centre and avg(d) as radius.

Step 2.1: For each and every circle determine the closest point, which is nearest to the circumference of each circle using min|(distance(r xi))|, where Xi takes the minimal distance from the circumference.

Step 2.2: Plot the points for each Pi, Xi as the Target Point Ti.

Step 2.3: Calculate the position of Ti as Ti(Pos) comparing it to the Pi for each particular circle.

Step 2.4: Find out the mode of Ti(Pos) i.e., find out maximum repeated Ti(Pos).

Step 2.5: If there is more than one mode then compute the mean of maximum repeated Ti(Pos)s.

Step 2.6: Mode of Ti(Pos) is the value of K which is the number of neighbourhood points.

Step 3: Compute the average of the distances of every point to all K of its nearest neighbors.

Step 4: Sort them in ascending order.

Step 5: Draw the averaged K-distances plot in an ascending order.

Step 6: The slope values are calculated at regular intervals.

Step 7: Compute difference between slope values at regular interval.

Step 8: If the difference is 10% higher than previous slope value then consider it as Epsi(i=1,2,…,n) value.

Step 9: ElseIf the difference is 25% higher than previous slope value then ignore it and consider it as a noise points.

Step 10: For each Epsi(i=1,2,…,n) value do,

Step 10.1: Calculate for each point in the dataset the number of data points in Eps neighbourhood.

Step 10.2: Compute MinPtsi(i=1,2,…,n) value for each Epsi(i=1,2,…,n) value by using the formula,

$$MinPtsi = \frac{1}{n\sum_{i=1}^{n} Pi}$$

where Pi(i=1,2,…,n) is the number of points in Eps neighbourhood of point i.

Step 11: For each Epsi(i=1,2,…,n) and MinPtsi(i=1,2,…,n) value,

Step 11.1: Assign Eps = Epsi and MinPts = MinPtsi.

Step 11.2: Adopt DBSCAN algorithm.

Step 11.3: plot the points in clusters related to Epsi as Ci.

Step 12: Repeat the step-11 and adopt DBSCAN algorithm with other Epsi and MinPtsi value ignoring the marked points.

Step 13: Finally the output displays all the obtained clusters(Ci) and non-marked points as outliers.

Figure 6: Pseudocode of Proposed KDTDBSCAN Algorithm

K-distance function performs mapping of each point to the distance d from its k^{th} nearest neighbour. Descending order of K-distance values and the graph of this function is related to the density distribution is done to sort the points. The graph obtained is called as sorted K-distance graph. Initially K is assigned with Eps value, dist(p) and the MinPts to k and the core points are those points which are equal or smaller than k. Ester et al., (1996) assigned K-distance graph for K value greater than 4 and it required more computations. The assignment of t to MinPts has given varied results and has be proved in several papers [6,22]. There for most researchers use the MinPts value as 4 during experimentations. The proposed DBSCAN algorithm is given in figure 6.The execution time needed for DBSCAN is calculated using O(nlog n) in which n is the size of the dataset being used. This formula is used in combined K-dist graph and DBSCAN procedure to calculate the execution time. The MinPts and Eps values are calculated automatically using this K- dist tree and so the execution time increases to O(n2 logn). O(log n) is the reduced value of the complexity of DBSCAN and the complexity for KDTDBCAN is given by O(knlog n).

5. EXPERIMENTAL RESULTS AND DISCUSSION

A Student database has been set up with 33 attributes and 2956 instances for evaluation of theproposed method. The student_mat.csv considered for evaluation has the following attributes like school, sex, age, address, traveltime, studytime, failures, schoolsup, famsup, famsize, famrel, freetime, goout, Pstatus, Medu, Fedu, Mjob, Fjob, reason, guardian, paid, activities, nursery, higher, internet, romantic, Dalc, G1, G2, G3, Walc, health and absences.

Parameter Setting

i. Data:datamatrixordist-object
   Method="dist"default=Euclideandistances
ii. eps=0.5
iii. MinPts=5
iv. scale=TRUE

Evaluation Measures

The performance of the research work is evaluated by using student dataset. The execution time is affected by dataset size and attributes size. To conclude with, some of the metrics used to evaluate the performance are Estimated number of clusters, Estimated number of noise points, Homogeneity, Completeness, V-measure, Adjusted Rand Index, Adjusted Mutual Information, Silhouette Coefficient, Speed of clustering and Memory usage. To get a better result, experiment has been performed 30 times and overall performance estimation was calculated by averaging the performance of the 30 individual runs. Using a single sample the Silhouette Coefficient is determined as:

$$S = \frac{b - a}{max\ (a, b)}$$

where, 'a' is the mean distance between a sample and all other points in the same cluster and 'b' is the distance between a sample and all other points in the next nearest cluster.

The measures mentioned in the table are used to evaluate the proposed KDTDBSCAN algorithm and compared with DBSCAN algorithm. From the work it clearly depicts that he traditional DBSCAN algorithm when used with KDT gives efficient results in terms of all the metrics.

S.No.	Parameters	DBSCAN	KDTDBSCAN
1	Estimated number ofclusters	3	3
2	Estimated number ofnoise points	18	21
3	Homogeneity	0.923	0.953
4	Completeness	0.743	0.883
5	V-measure	0.821	0.917
6	AdjustedRandIndex	0.841	0.952
7	Adjusted MutualInformation	0.901	0.916
8	SilhouetteCoefficient	0.626	0.7809
9	Total running(execution)time	0 mins. 0.431secs.	0 mins. 0.442311secs.

10	Estimated memoryusage	9MB	7.5MB

Table 1: Result obtained for DBSCAN and KDTDBSCAN Algorithms

When the distance function and eps values are related , the results shows major impact which can be noted from the results obtained. The distance function and MinPts are appropriately chosen at random. So there is a great variation in the result.

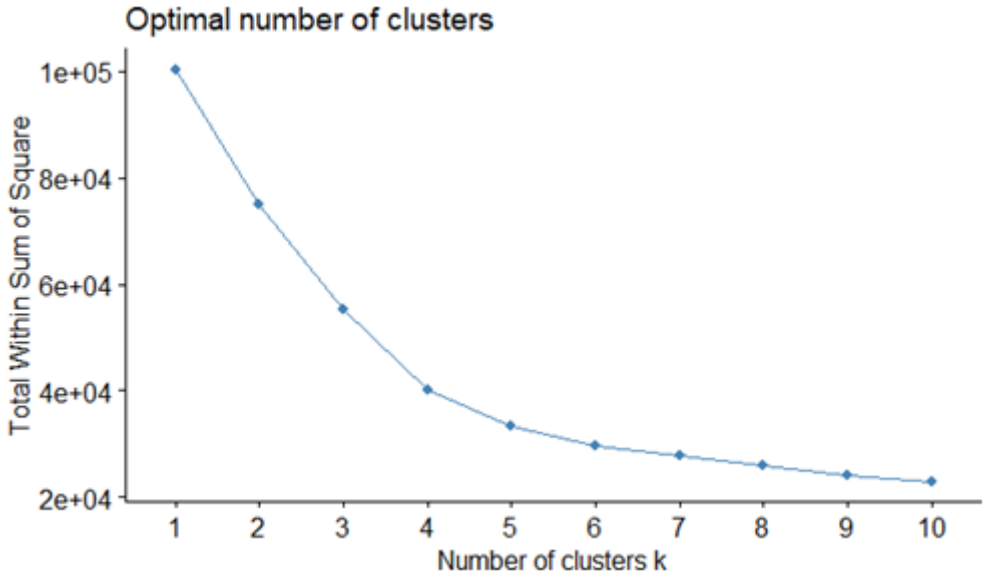

Figure 7: Sum of square value for different 'k' values

The execution time and the silhouette coefficient is estimated and compared with the traditional DBSCAN algorithm. The graph shown the optimal number of clusters that can be obtained when the k-distance value is changed automatically is given in figure7.

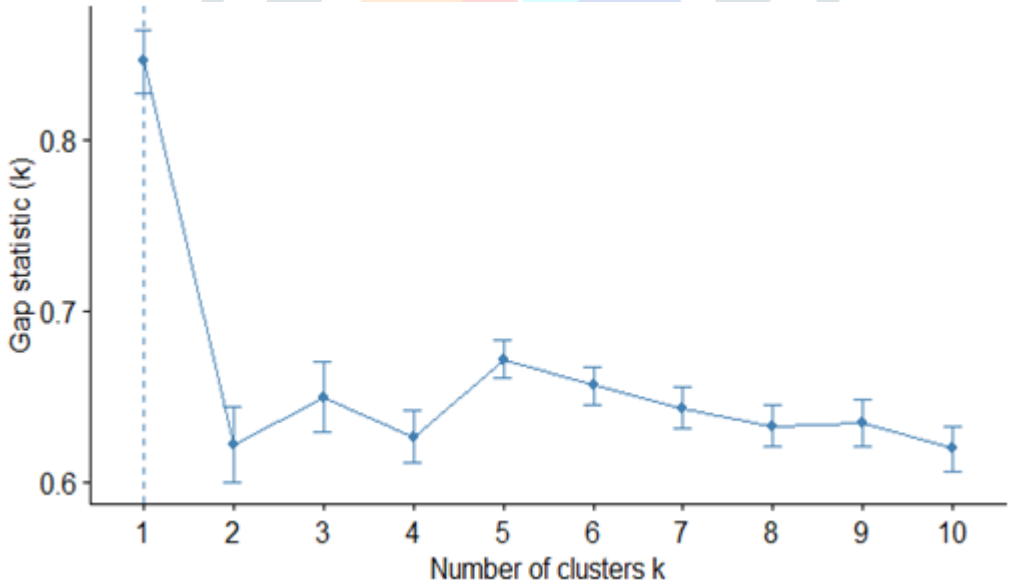

Figure 8: Gap statistics for different 'k' values

The gap statistic obtained for different values of k is shown in figure 8. The graph showing the execution time and silhouette measure is given in figure 9.

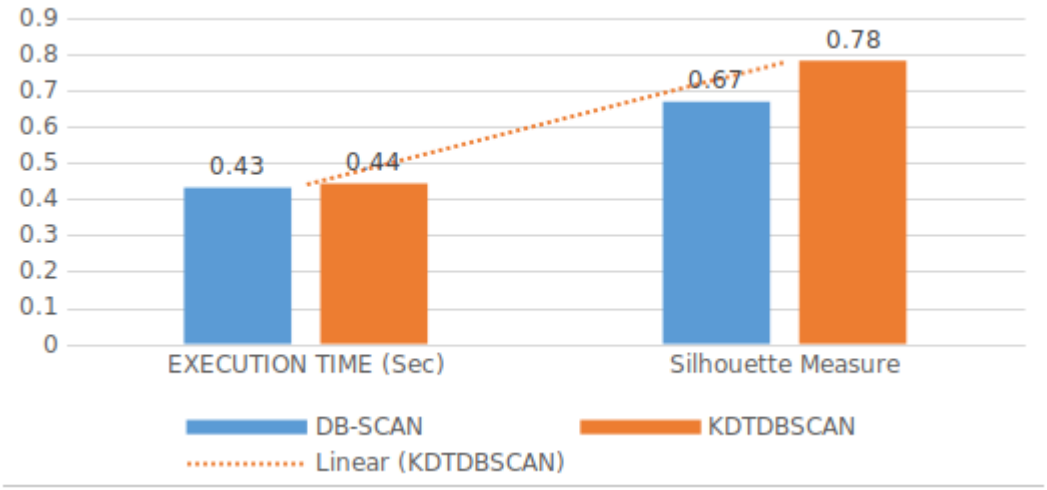

Figure 9: Performance of DBSCAN and KDTDBSCAN Algorithms

6. CONCLUSION AND FUTURE WORK

In this paper the performance evaluation of a proposed KD-Tree based DBSCAN clustering algorithm is established. Also logically compares the features of K-Distance tree based DBSCAN and traditional DBSCAN clustering algorithms. It also compares the performance of these two algorithms when they are applied on educational databases. The result obtained shows that automatic selection of epsilon and MinPts using KD-Tree based method has introduced drastic change in the performance of DBSCAN algorithm in terms of time silohettute coefficient and other metrics.

In future plan to modify HDBSCAN algorithm which can execute faster than OPTICS and can flat partition the database and the most prominent clusters can be extracted from the hierarchy.

REFERENCES

[1] Anil K Jain, M.N.Murty and P.J.Flynn, 1999. Data Clustering: A Review, ACM Computing Surveys, 31(3): 264-323.

[2] B. Borah and D. K. Bhattacharyya, 2004. An improved sampling-based DBSCAN for largespatial databases, In Proc. International Conference on Intelligent Sensing and InformationProcessing,pp: 92-96.

[3] Campello, R.J.G.B., Moulavi, D., Zimek, A. and Sander, J., 2013. A framework for semi-supervised and unsupervised optimal extraction of clusters from hierarchies. Data Mining and Knowledge Discovery, 27: 344-371.

[4] Cheng-Fa Tsai and Chun-Yi Sung, 2010. EIDBSCAN: An Extended Improving DBSCAN algorithm with sampling techniques, International Journal of Business Intelligence and DataMining, 5(1): 94-111.

[5] Chih-Ping Wei, Yen-Hsien Lee and Che-Ming Hsu, 2000. Empirical comparison of fastclustering algorithms for large data sets, In Proc. 33rd Annual Hawaii International Conference on System Sciences, IEEE Explore Digital Library, pp:1-10.

[6] Dinh Phung, Brett Adams, Kha Tran, Svetha Venkatesh and Mohan Kumar, 2009. High accuracy context recovery using clustering mechanisms, In Proc. 7th IEEE International Conference on Pervasive Computing and Communications, pp: 122-130.

[7] Domenica Arliaand Massimo Coppola, 2001. Experiments in Parallel Clustering with DBSCAN, Springer-Verlag, LNCS 2150, pp: 326–331.

[8] Erich Schubert, Jörg Sander, Martin Ester, Hans-Peter Kriegeland Xiaowei Xu, 2017. DBSCAN Revisited: Why and How You Should (Still) Use DBSCAN. ACM Transaction Database Systems. 42(3): 1-21.

[9] Erich Schubert, Sibylle Hess and Katharina Morik, 2018. The Relationship of DBSCAN to Matrix Factorization and Spectral Clustering, Lernen, Wissen, Daten, Analysen (LWDA), pp:330-334.

[10] Hans-Peter Kriegel, Erich Schubert and Arthur Zimek, 2017. The (black) art of run time evaluation: Are we comparing algorithms or implementations?, Knowledge and Information Systems,52: 341-378.

[11] Hans-Peter Kriegel, Peer Kröger, Jörg Sander and ArthurZimek, 2011. Density-based Clustering, WIREs Data Mining and Knowledge Discovery, Wiley Online Library, 1(3): 231-240.

[12] Hencil J Peterand A, Antonysamy, 2010. An Optimised Density Based Clustering Algorithm,

International Journal of Computer Applications, 6(9): 20-25.

[13] Hua Jiang, Jing Li, Shenghe Yi, Xiangyang Wang and Xin Hu, 2011. A new hybrid method based on partitioning-based DBSCAN and ant clustering, Expert Systems with Applications: An International Journal,Elsevier,38(8): 9373-9381.

[14] Ivancsy Renata and Kovacs Ferenc, 2006. Clustering techniques utilized in webusage mining, In Proc. 5th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and DataBases (AIKED'06), pp: 237–242.

[15] Jian Li. Wei Yu and Bao-PingYan, 2009. Memory effect in DBSCAN algorithm, In Proc.4th International Conference on Computer Science & Education, pp 31-36.

[16] Jiang Sheng-Yi and Li Xia, 2009. A Hybrid Clustering Algorithm, In Proc. 6th International Conference on Fuzzy Systems and Knowledge Discovery, IEEE Computer Socity, 1:366-370.

[17] Jon Louis Bentley, 1975. Multidimensional Binary Search Trees used for Associative Searching, Communications of the ACM, 18(9): 509-517.

[18] Jörg Sander, 1998. Generalized Density-Based Clustering for Spatial Data Mining, Dissertation, München: Herbert Utz Verlag, ISBN:3-89675-469-6, pp:1-226.

[19] Jörg Sander, Martin Ester, Hans-Peter Kriegel and Xiaowei Xu, 1998. Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications, Data Mining and Knowledge Discovery, 2:169-194.

[20] Ling R.F, 1972. On the theory and construction of k-clusters, The Computer Journal, 15(4):326-332.

[21] Martin Ester, Hans-Peter Kriegel, Jiirg Sanderand XiaoweiXu, 1996. A density-based algorithm for discovering clusters in large spatial databases with noise, In Proc.2nd International Conference on Knowledge Discovery and Data Mining (KDD-96), Spatial, Text and Multimedia, AAAI Press, pp: 226-231.

[22] Raiser Stefan, Edwin Lughofer, Christian Eitzinger and James Edward Smith, 2010. Impact of object extraction methods on classification performance in surface inspection systems, Special Issue Paper, Machine Vision and Applications, Springer-Verlag, 21, 627–641.

[23] RamBhakare, Ketki, Krishna, R and Bhakare, Samiksha, 2012. An Energy-Efficient Gridbased Clustering Topology for a Wireless Sensor Network. International Journal of Computer Applications. 39(14): 24-28.

[24] Ricardo J.G.B. Campello, Davoud Moulavi, Arthur Zimek, and Jorg Sander, 2015. Hierarchical density estimates for data clustering, visualization, and outlier detection, ACM Transactions on Knowledge Discovery from Data, Article 5, 10(1): 1-51.

[25] Sathiamoorthy Jayaraman, Ramakrishnan Bhagavathiperumal and Usha Mohanakrishnan, 2018. A Three Layered Peer-to-Peer Energy Efficient Protocol for Reliable and Secure Data Transmission in EAACK MANETs, Springer Journal of Wireless Personal Communication, 102, 201-227.

[26] Sathiamoorthy, J and Ramakrishnan, B, 2017. A Reliable Data Transmission in EAACK MANETs using Hybrid Three-Tier Competent Fuzzy Cluster Algorithm, Springer Journal of Wireless Personal Communication, 97, 5897-5916.

[27] Sathiamoorthy, J and Ramakrishnan, B, 2018. A competent three-tier fuzzy cluster algorithm for enhanced data transmission in cluster EAACK MANETs, Springer Journal of Soft Computing, 22, 6545–6565.

[28] Sathiamoorthy, J, Ramakrishnan, B and Usha, M, 2015. A reliable and secure data transmission in CEAACK MANETs using Distinct Dynamic Key with classified Digital Signature Cryptographic Algorithm, IEEE International Conference on Computing and Communications Technologies (ICCCT), pp.144-151.

[29] Ulrich Bayer, Paolo MilaniComparetti, Clemens Hlauschek, Christopher Kruegel and Engin Kirda, 2009. Scalable, Behavior-Based Malware Clustering, In Proc. 16th Annual Network and Distributed System Security Symposium (NDSSS 2009), pp: 1-18.

[30] Usha Mohanakrishnan, Ramakrishnan Bhagavathiperumal and Sathiamoorthy Jayaraman, 2018. A Trusted Waterfall Framework Based Peer to Peer Protocol for Reliable and Energy Efficient Data Transmission in MANETs, Springer Journal of Wireless Personal Communication, 102, 95–124.

[31] Viswanath Pulabaigari and Pinkesh Rajwala, 2006. l-DBSCAN: A Fast Hybrid Density Based Clustering Method, In Proc. 18th IEEE Computer Society Conference on Computer Vision and Pattern Recognition(CVPR), pp: 912-915.

[32] Y.El-Sonbaty, M.A.Ismail and M.Farouk, 2004. An efficient density based clustering algorithm for large databases, In Proc. 16th IEEE International Conference on Tools with Artificial Intelligence,

pp: 673-677.

[33] Yang Fan and Rao Yutai, 2011. A Density-based Path Clustering Algorithm, In Proc. IEEE International Conference on Intelligent Computation and Bio-Medical Instrumentation, pp:224-227.

[34] Yiqun Cao, Tao Jiang and Thomas Girke, 2010. Accelerated similarity searching and clustering of large compound sets by geometric embedding and locality sensitive hashing. Bioinformatics, 26(7): 953-959.

[35] Zhou Shuigeng, Zhou Aoying, Jin Wen, Fan Ye and Qian Weining, 2000. FDBSCAN: A fast DBSCAN algorithm, Journal of Software, 11, 735-744.