Journal of Computer Science

Original Research Paper

Real Time Density-Based Clustering (RTDBC) Algorithm for Big Data

Dr. B. Ravi Prasad

Professor, Department of CSE, Marri Laxman Reddy Institute of Technology & Management, Dundigal (V&M), Medchal Dist., Hyderabad, Telangana State India

Article history
Received: 28-06-2017
Revised: 21-07-2017
Accepted: 10-08-2017

Email: rprasad.boddu2017@gmail.com

Abstract: Density Based Spatial Clustering of Applications with Noise (DBSCAN), a well-known Density-Based Clustering Algorithm is a advanced data clustering method with various applications in numerous fields like Satellites images, X-ray crystallography, Anomaly Detection in Temperature Data. But its run time $R(n^2)$ complexity draws a major challenge. In this research paper, we propose a new unique algorithm called Real Time Density Based Clustering RTDBC to minimize the problems in DBSCAN. In proposed algorithms, objects are allotted into clusters using labels representatives than the method of propagating directly to reduce propagation time of label considerably. In contrast, RTDBC produce fast result and continuous process of runtime and additionally users are permitted to suspend for testing the result and continue as to enhance good results.

Keywords: DBSCAN, RTDBC, Data Clustering, Big Data

Introduction

DBSCAN, a familiar density based data clustering algorithm introduced by Easter *et al*. (1996). It has a fast solution for complicated clusters assigned one input parameter and suggested the value of the parameter for user. In huge data bases it was 1900 times faster and expected improved final results ended up. DBSCAN identify the clusters which are in arbitrary shape and also for finding outliers. A set of Dense objects connected and separated by a new created cluster with low density region clusters while density object more than p objects inside ε radius of neighborhood. DBSCAN is mainly considerable clustering algorithm with various applications and extension (Brecheisen *et al*., 2004; Gan and Tao, 2015) like satellite images, x-ray crystallography, anomaly detection in temperature data, astronomy (Settles, 2009) and neuroscience (Mai *et al*., 2012). Bur its real weakness complex data sets if they located too close with each other even if they are different densities.

During cluster extension process DBSCAN (Ester *et al*., 1996) executes and determines the ε-radius of the neighborhood queries of all objects for data grouping. Thus, it has two sources:

- Range of n query process, ensuring as $R(d_x n^2)$ where d_x = worst case complexity of distance, n = no. of objects
- Propagation process of label with $R(d_x n^2)$: it's a time complexity to allocate objects as labels.

These two sources rapidly turn to block while increasing the volume and aims for various works for improving DBSCAN. These techniques (both sources) results to accelerate of DBSCAN by either of two sources means without the data information exploited the improved performance. A filter (Fast Lower Bound) by source 2, the calculations of the true distance reduced along with increased label propagation time to maintain the order of initial list. The Data space divides by grids in Grid Based Technique (GBT) for saving the run time as each cell perform locally.

Information of the data is not utilized considerably by all these techniques, they earn excessive distance computation, thus tends to limit the performance efficiency. Hence, we propose a new unique algorithm called Real Time Density Based Clustering RTDBC to minimize the problems in DBSCAN.

When compare to previous techniques, it upholds accurate results always and reads the current structure data then it considers a small object into subsets for refining all iterations. So, it replace the label propagating directly with objects are in to clusters by the representative of the labels. Hence propagation time of the label is decreased considerably.

During execution in existing approaches works on batch scheme, does not permit the user communication (Brecheisen *et al*., 2004; Gan and Tao, 2015). In other side, during the run time anytime algorithm (Zhou *et al*., 2000) rapidly generate approximate result and refine continuously and permit the users to suspend for verifying the result, resume to finding satisfactory result is obtained. So, RTDBC algorithms have suitable method and broadly applied for various areas (Zhou *et al*., 2000). But all existing methods are designed mainly for small datasets due to their space complexity and high time. Hence, proposed RTDBS algorithm aiming to provide for very large datasets.

There are very few algorithms works on complex data like images and graphs (Brecheisen *et al*., 2004) but facing a scalability problems. But in proposed RTDBC algorithm effectively works on very large complex data and minimize the high time and space complexity.

In this research paper, a proposed new RTDBC algorithm for clustering very large complex datasets that represents all described problems above. The proposed RTDBC algorithm ahead with the advantages:

- RTDBC dynamically study the information of data and apply to decrease the propagation time of the label with number of range queries. So, it is considerably speed up the runtime to extent level of magnitude compared with DBSCAN and other approaches
- RTDBC runs initial runtime very low for better results compared and user interaction for to get good considerations in arbitrary time
- RTDBC useful for clustering very large complex datasets

Related Work

Density based Data Clustering Algorithm

Definition 1: ε-Neighborhood

Figure 1 describes ε-neighborhood of objects within the radius of ε from an object, the ε neighborhood of an object p represented by $N_\varepsilon(p)$ then:

$$N_\varepsilon(p):\{q \mid d(p.q) \le \varepsilon\}$$

Definition 2: High Density

Figure 2 describes ε-Neighborhood of an object contains at least MinPts of objects.

Definition 3

Figure 3 describes core, Border and Outlier (Noise).

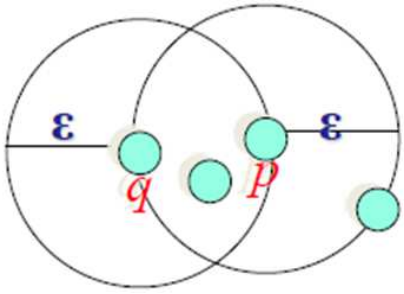

Fig. 1. ε-Neighborhood of *p*; ε-Neighborhood of *q*; MinPts = 4 (Density of *p* is "high"); MinPts = 4 (Density of *q* is "low")

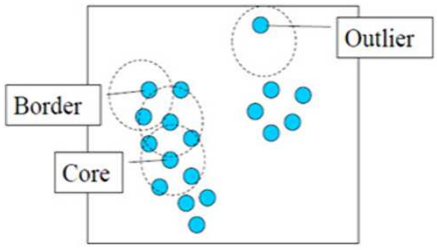

Fig. 2. MinPts = 5, ε = 1unit; Core: core points are at inside of the cluster and it has more than a specified number of points (MinPts) within ε; Border: A border point has fewer than MinPts within ε, but is in the neighborhood of a core point; Outlier (Noise): A Outlier (Noise) point is any point that is not a core point nor a border point

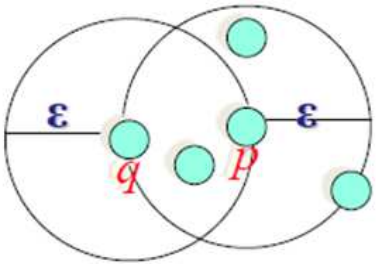

Fig. 3. MinPts = 4; *q* is directly density reachable from *p*; *p* is not directly density reachable from *q*

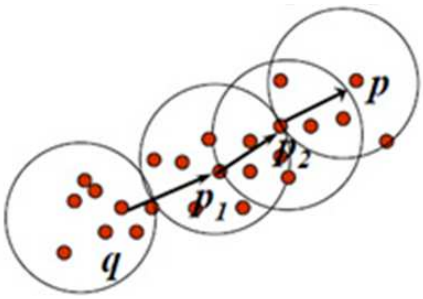

Fig. 4. *MinPts* = 7; *p* ←*p*2 ←*p*1← *q* form a chain; *p* is (indirectly) density reachable from *q*; *q* is not density reachable from *p*

Definition 4

Density Reachability

Figure 4 describes asymmetric object q is directly density-reachable from object p if p is a core object and q is in p's ε -neighborhood

Density Connectivity:

- Point p is directly density-reachable from $p2$
- $p2$ is directly density-reachable from $p1$
- $p1$ is directly density-reachable from q

DBSCAN Algorithm

Algorithm1

for each o ϵ D do
if o is not yet classified then
if o is a core-object then
collect all objects density reachable from o
and assign them to a new cluster
else assign o to NOISE

DBSCAN arbitrarily draws object p (unlabelled) and executed q ϵ N_ε (p) while p is core object, then objects are labeled for p including all density connected objects of p.

Proposed Algorithm: RTDBC (Real Time Density Based Clustering)

RTDBC algorithm is a solution for time consuming in many areas like object recognition (Kobayashi *et al*., 2013) and robotics (Zhou *et al*., 2000). The main idea of this algorithm is to produce approximate results immediately and continuously drawing the results till to extract the acceptable results or solutions. This algorithm also analyzes the intermediate results on interruption while running and resumed for extract acceptable solutions this representation is shown in Fig. 5.

Figure 6 shows the development of different algorithms of proposed RTDBC and observed that the performances of a (Zhou *et al*., 2000) is better quality than others (B, C). Hence A preferred for many works for better solution and other side C stands on worst performance.

The main approach of proposed RTDBC algorithm is shown in Fig. 7.

By illustrating the Fig. 7, C1 cluster is determined completely while select the two objects f, g then:

- C1 and C2 are the final Clusters
- Two small clusters are formed inside C1, by a, b and with their neighbors
- Two more small clusters form inside C2 by d, e and with their neighbors
- Outlier is c
- a, b are density connected together while core object is f
- border object g permits to find the core object h without performing query as h having minimum μ neighbors
- C2 also determined with d and e which are density connected together.

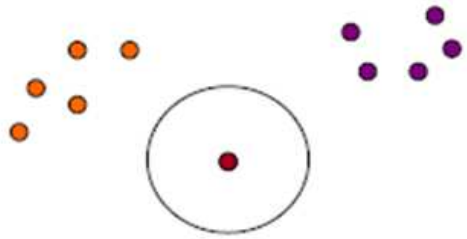

Fig. 5. *MinPts* = 3, **ε** = 2 cm

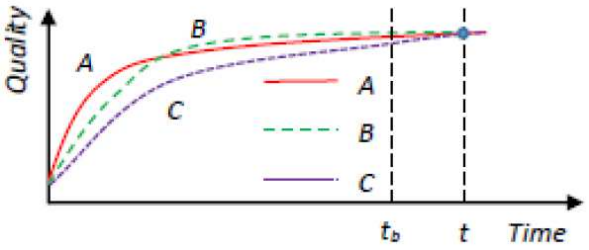

Fig. 6. Performance of different algorithms of RTDBC (t_b = Runtime)

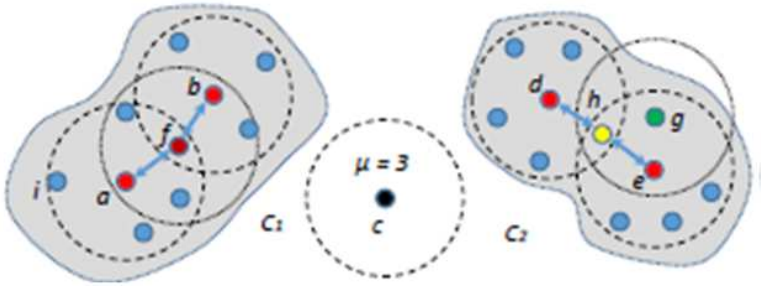

Fig. 7. Proposed RTDBC algorithm

Hence the proposed RTDBC extract the same results as in DBSCAN without executing all queries, result that time reduced in clustering.

The pseudo code RTDBC algorithm shown in algorithm 2 described in nine major steps.

Step 1: Design a Structure of an initial cluster
Step 2: Developing the cluster graph as G = (V, E)
Step 3: Identifying the connected components
Step 4: Merging the connected components
Step 5: Verifying a stopping condition
Step 6: Choosing objects for queries
Step 7: Activating queries
Step 8: Updating cluster graph

Algorithm2: Pseudo Code of RTDBC Algorithm

function C = RTDBClu (*R, ε, μ, α, β, d)*
input: *O* dataset, *ε, μ*, parameters, d function of distance DBSCAN
α, β block size of the query
output: *C* the result of final clustering
begin
/* *step 1: Design a Structure of an initial cluster* */
while there exist objects untouched in *R do*
S = set of *α untouched* objects
for all objects *o* in *S do*
perform range query on *o* and mark the state of *o*
if o is a core object *then* mark the states of its neighbors in *Nε*(*o*)
if o is a noise object *then* put *o* and $N_\varepsilon(o)$ into the noise list *L*
/* *step 2: developing the cluster graph as G = (V, E)*/
put all primitive clusters into *V* as nodes
determine the states of all edges *e* in *E*
/* *repeatedly select objects for range queries until terminated* */
do
/* *step 3: identifying the connected components* */
find all connected components of *G* via the *yes* states
/* *step 4: merging the connected components* */
merge each connected component of *G* into a single node
calculate the state of each edge of the new graph *G*
return an intermediate clustering result *C'* if required
/* *step 5: verifying a stopping condition* */
b = check if *G* only contains edges with *yes* or *no* states
if b = false then
/* *step 6: choosing objects for queries* */
for all nodes *v* in *V do*
calculate the node statistic for *v*
calculate the node degree for *v*
calculate object scores for all unprocessed objects in *O*
S = set of *β* objects with highest scores
/* *step 7: activating queries* */
for all objects *o* in *S do*
perform range queries on the object *o*
update the states of *o* and its neighbors *Nε*(*o*)
merge $N_\varepsilon(o)$ to all nodes that contain *o*
/* *step 8: updating cluster graph* */
update the states of all edges *e* in *E*
while the stopping condition is not reached (*b* = *false*)
/* *step 9: processing the outliers* */
for all objects *o* in *L do*
check if *o* is truly a noise or a border object
return the final clustering result *C*

In step 1 RTDBC queries objects *α* in size of blocks and β for step 6 to 7. Hence selection of objects α and *β* for activating queries for all iterations of step 1 and 6, 7 as to provide main benefits;

- The quality of intermediate clustering at earlier steps been enhancing with overlapping of primitive circles
- Anytime scheme of the overall overhead been reducing as by using $\alpha = \beta$

Assume that RTDBC is run at the end; its end results are absolutely identical from DBSCAN.

Here we analyze a RTDBC algorithm of worst case complexity. Lets assume:

- Number of objects = *n*
- Number of G initial nodes of *G*, $|V| = v$
- Number of nodes at iteration $i = v_i$
- Number of nodes at iteration $v_0 = v = v_i$
- Noise list size *L*, $|L| = l$
- Number of RTDBC update iterations = *b*

Therefore:

Step 1: Time required for querying and initializing the objects = *R*(*vn*)
Step 2: Time required for developing the structure graph = $R\ (v^2 n)$
Step 3: Time required for identifying the connected components = $R = \left(\sum_{i=1}^{b} v_i^2 - 1\right)$
Step 4: Time required for merging the connected components = $R = \left(\sum_{i=1}^{b} v_{i-1^n}\right)$
Step 5: Time required for relabeling the edges states = $R = \left(\sum_{i=1}^{b} v_i^2\right)$

Time required for updating the unprocessed objects for all inside nodes = $R = \left(\sum_{i=1}^{b} v_i\, n\right)$

Time required for verifying a stopping condition = $O = \left(\sum_{i=1}^{b} v_i^2\right)$

Step 6: Time required for calculating degrees of node = $R=\left(\sum_{i=1}^{b} v_i^2\right)$

Time required for calculating score of the object = $R\left(\sum_{i=1}^{b} v_i\left(n-v-(i-1)\beta\right)\right)$

Time required for sorting objects unprocessed = $R\left(\sum_{i=1}^{b}\left(n-v-(i-1)\beta\right)\right)\log\left(n\text{-}v\text{-}(i\text{-}1)\beta\right)$

Step 7: Time required for querying = $R\ (b\beta n)$

Time required for merging = $R\left(\sum_{i=1}^{b} v_i \beta n\right)$

Time required for updating the size of the node = $R\left(\sum_{i=1}^{b} v_i n\right)$

Step 8: Time required for updating the cluster graph = $R\left(\sum_{i=1}^{b} v_i^2 n\right)$

Step9: Time required for processing the outliers = R (l μ,n)

The real time complexities in RTDBC are very smaller than those illustrated above and consideration of experimental analysis. Therefore:

- The maximum iterations in RTDBS = $v_i >> v >> n$ *and* $b>>b_{max}$, where $b_{\max} = (n\text{-}v)/\beta$ and
- The run time complexity of RTDBS $O(n^2)$ very smaller than DBSCAN

So, RTDBC requires:

- Space for storing the graph $G = R\ (v^2 + v_n + n + v + l\mu)$
- The space complexity of RTDBC in the worst case = $R\ (n^2)$…… $v >> n$

Experimental Results

We create larger data sets of 2D- 4 synthetic DS1 to DS4 data sets having 16 to 32 clusters, contains 3254-9554 points which are placed randomly

DS1 data set added 99 more objects which are placed additionally to the original data sets for all objects (DS1x100) for analyzing of arbitrarily clusters in RTDBC. We also study the characteristics of RTDBC on increasing the number of objects while maintained the cluster structure.

We use $\alpha = \beta = 512$
$\mu = 5$,
$\epsilon = 1$

The performance of RTDBC is shown in Fig. 8 by increasing objects for DS1 to DS4. It is observed that in Fig. 8b, RTDBC significantly faster than DBSCAN. It means denser of the clusters, speedup factors are high and the solutions are found in Fig 8c and 8d.

First, RTDBC used very few queries compared to DBSCAN. Therefore, it needs only 0.25% (6964.4) range queries for clustering DS1x300 on average with objects of 2783567. Second, graph nodes initial numbers of are much small, 0.12% (3441.6) for clustering on average DS1x300.

However, the graph nodes are considerably reduced during runtime on all iterations shown in Fig. 8e and the time of label propagation also reduced. Thus the RTDBC is significantly the faster at the end than the DBSCAN.

Normalized - Mutual Information (NMI) used for extracting the results of intermediate clustering and compare real results. If the Result is perfect clustering means 1 and respectively. The results of perfect clustering is shown in Fig. 9 even at first step with high scores.

NMI of the DBSCAN is 0.009 where RTDBC starts with 0.998. It is noted that RTDBC requires only 4.4 sec and the DBSCAN needs 252.4 sec in DS1x0200 (1855767). It means RTDBC is 57.3 times faster than DBSCAN i.e., it gives efficient method for very large data sets Illustrate the Fig. 10, RTDBC cumulative run time and NMI on selecting objects and random method in DS1x0200:

For selecting objects in Step 6

- Number of iterations: 7
- Time required: 8.3 sec
- Number of queries: 6149

For random objects in step 6

- Number of iterations: 225
- Time required on selecting objects: 87.7
- Number of queries: 117761

The initial nodes in G increases with respect to α because the primitive circles are overlapped in step 1 shown in Fig. 11b and due to merging in step 4 leads to faster reduction in graph nodes. The nodes of the graph also decreases more rapidly on all iterations shown in Fig. 11d. Hence the RTDBC cumulative run times reduced considerably shown in Fig. 11a. Hence the numbers of states edges identification required more queries and increased queries are stable while β is large. Therefore, more or additional core objects are identified on each step and thus making to rapid detection the "yes" states of the edges shown in Fig. 11c. So the increased queries effect is very small on operation cost and the cumative run times of RTDBC are still reduced. Overall performance decreases from redundant queries while α, β are very large. Hence in RTDBC prefer the method that for maximum iterations while $\alpha = \beta$.

DOI: 10.3844/jcssp.2017.496.504

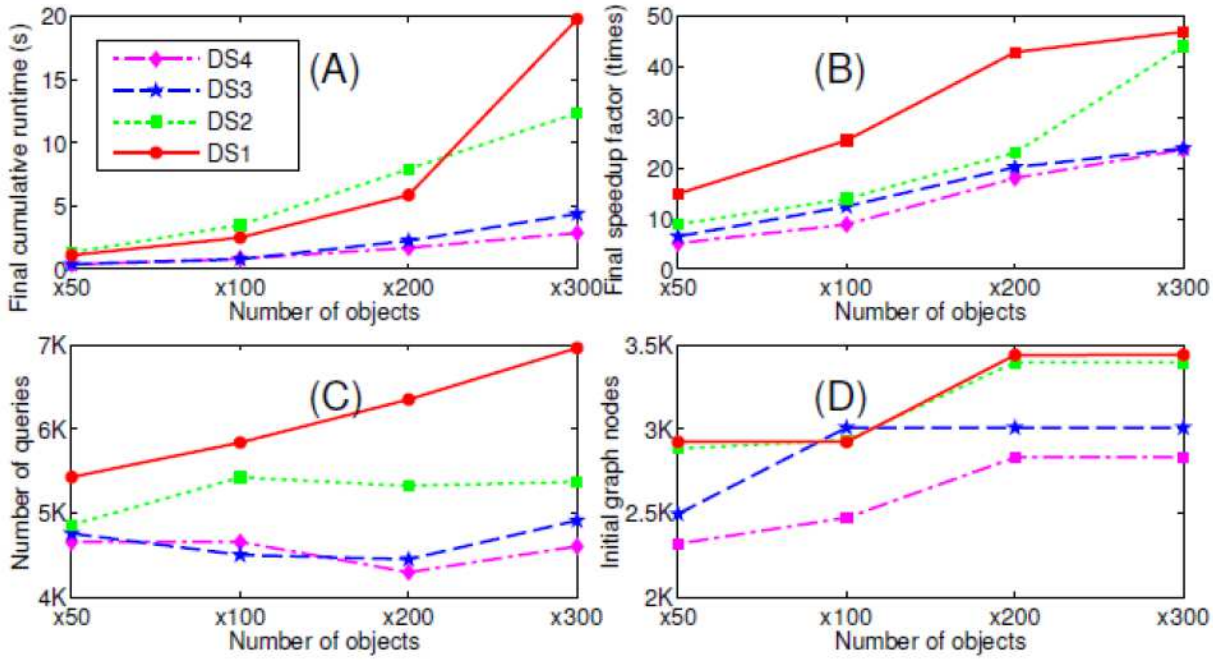

Fig. 8. The performance of RTDBC

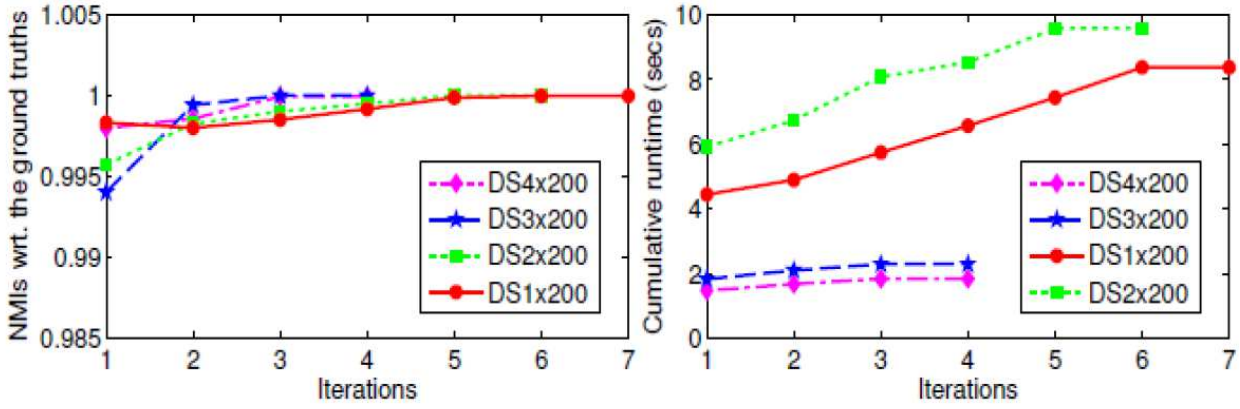

Fig. 9. NMI results

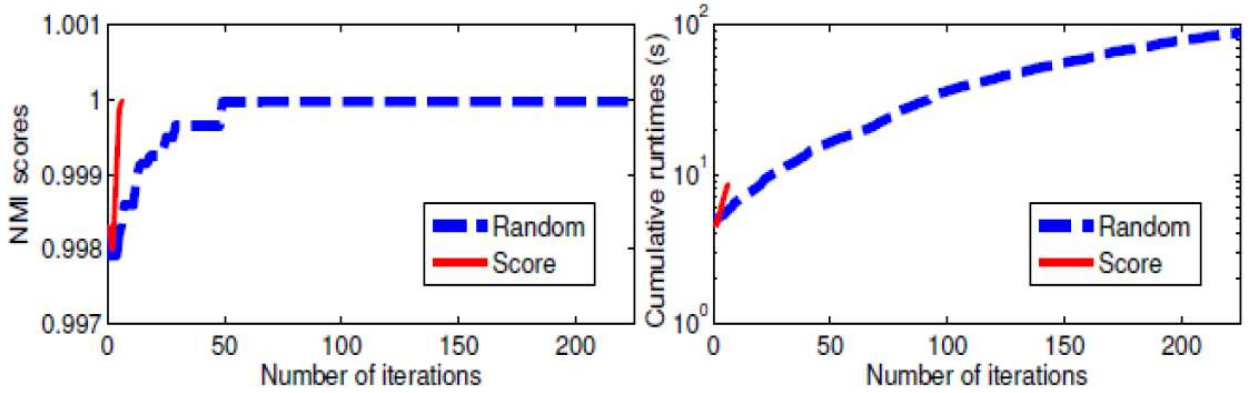

Fig. 10. Performance of RTDBC for DS1x0200 (active selection)

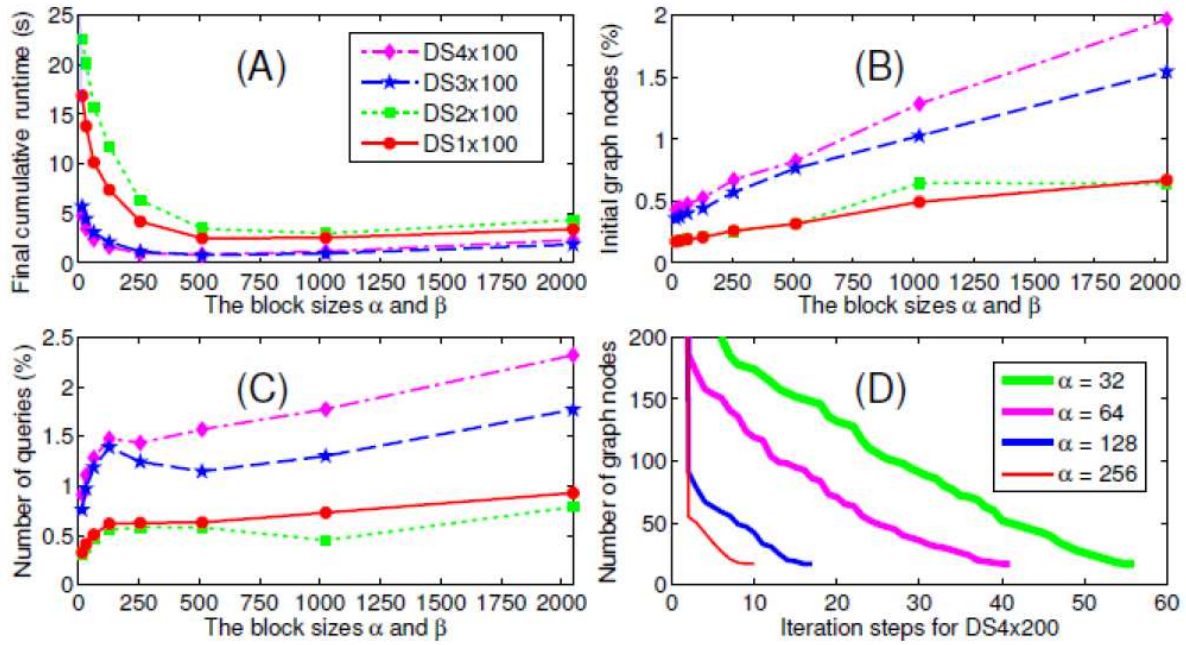

Fig. 11. Role of α and β

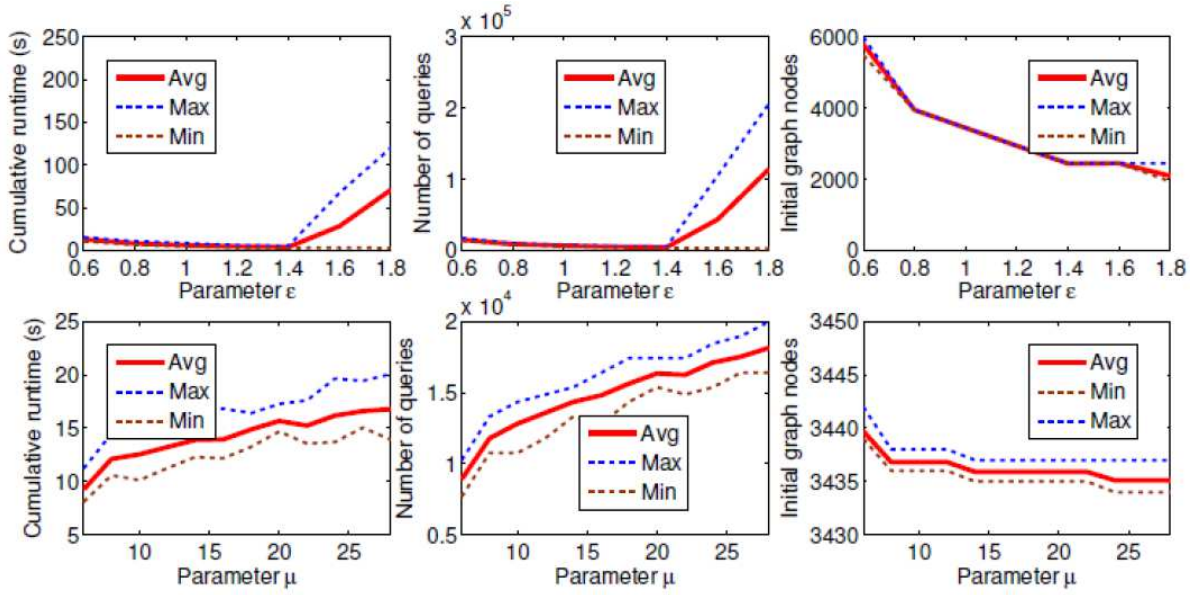

Fig. 12. Effect on RTDBC performance by μ and ϵ

The RTDBC runtime slightly increases by increasing the parameter μ and more queries needed to find unprocessed objects shown in Fig. 12. Thus the graph size decreases tends to reduce the cost. In other hand this is happen while noise objects are more.

Increased the value of ϵ will impact to decrease the initial graph nodes while more objects are labeled inside the primitive circle. However, number of queries and runtimes are decreased. Thus, the larger of ϵ tends RTDBC to obtain faster clustering results of all iterations.

RTDBC performance on different synthetic datasets created by DBSCANR (DBSCAN variant) (Gan and Tao, 2015) shown in Fig. 13 with synthetic 1 (9 with 2000000 points) and synthetic 2 (11 with 2000000 points) dimensions on different values of ϵ with μ = 5. It is observed that the performance of RTDBC very faster compared to DBSCAN and its variant DBSCANR.

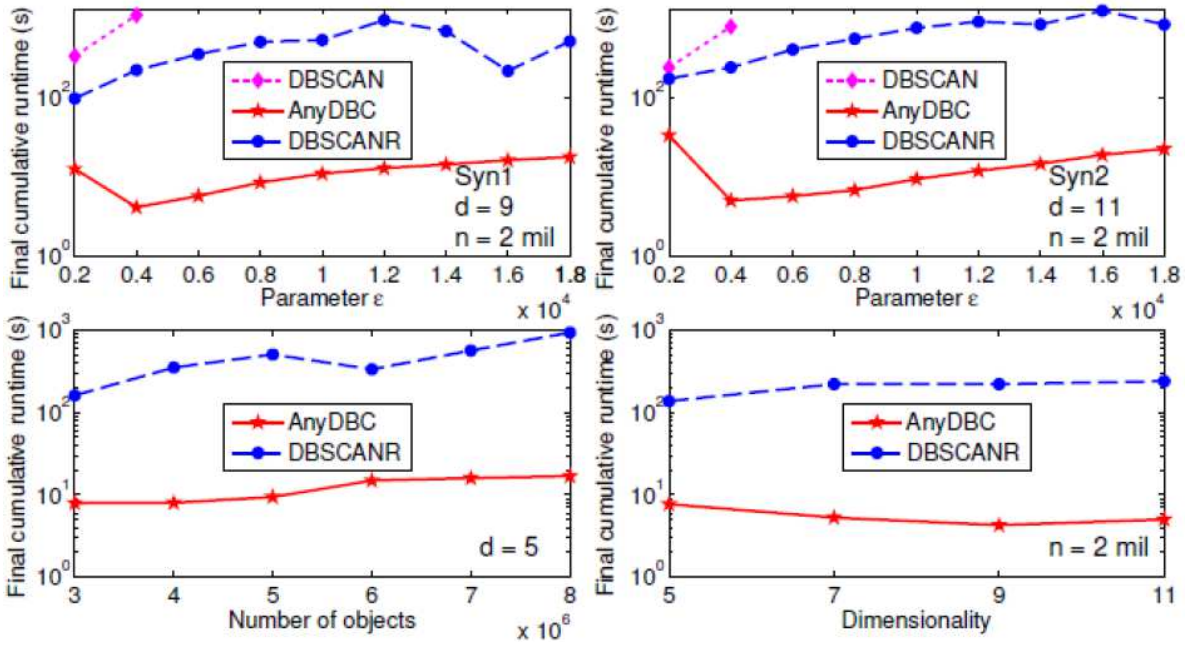

Fig. 13. RTDBC performance on different synthetic datasets

It is also noted that in Synthetic 1 data set while ϵ = 4000 RTDBC needs 3.68 sec where DBSCAN and DBSCANR needs 1093.6 sec and 221 sec respectively. Thus, RTDBC is 297.1 times faster than DBSCAN and 60 from DBSCANR.

Scalability of RTDBC with respect to DBSCANR shown in Fig. 13b with μ = 5 and ϵ = 5000 and μ = 5 and ϵ = 4000 of number of objects and data dimension respectively. It is noted that the efficient performance of RTDBC on higher values of objects and data dimension. Thus, for clustering of 5000000 objects RTDBC completes in bellow 9.3 sec where as 505.4 sec and 19388.8 sec in DBSCANR and DBSCAN respectively. However, overall RTDBC is nearly 55.5 faster compared to DBSCANR and DBSCAN.

Conclusion

Though DBSCAN, a well-known Density-Based Clustering Algorithm is a advanced data clustering method with various applications in numerous fields, but its run time $R(n^2)$ complexity draws a major challenge. RTDBC is a solution to minimize the problems in DBSCAN. In RTDBC objects are allotted into clusters using labels representatives than the method of propagating directly to reduce propagation time of label considerably. In contrast, RTDBC produce fast result and continuous process of runtime and additionally users are permitted to suspend for testing the result and continue as to enhance good results. RTDBC is 297.1 times faster than DBSCAN and 60 from DBSCANR. Clustering of 5000000 objects RTDBC completes in bellow 9.3 sec where as 505.4 sec and 19388.8 sec in DBSCANR and DBSCAN respectively. However, overall RTDBC is nearly 55.5 faster compared to DBSCANR and DBSCAN.

Acknowledgment

The author expresses his appreciation of the Mrs Battula Sridevi to his valuable helps in this research.

Ethics

This article is original and contains unpublished materials. The corresponding author confirms that all of the other authors have read and approved the manuscript and there are no ethical issues involved.

References

Brecheisen, S., H. Kriegel and M. Pfeie, 2004. Efficient density-based clustering of complex objects. Proceedings of the 4th IEEE International Conference on Data Mining, Nov. 1-4, IEEE Xplore Press, Brighton, UK, pp: 43-50. DOI: 10.1109/ICDM.2004.10082

Ester, M., H.P. Kriegel, J. Sander and X. Xu, 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, Aug. 02-04, AAAI Press, Portland, Oregon, pp: 226-231.

Gan, J. and Y. Tao, 2015. DBSCAN revisited: Mis-claim, un-fixability and approximation. Proceedings of the ACM SIGMOD International Conference on Management of Data, May 31-Jun. 04, ACM, Melbourne, pp: 519-530. DOI: 10.1145/2723372.2737792

Kobayashi, T., M. Iwamura, T. Matsuda and K. Kise, 2013. An anytime algorithm for camera-based character recognition. Proceedings of the 12th International Conference on Document Analysis and Recognition, Aug. 25-28, IEEE Xplore Press, Washington, DC, USA, pp: 1140-1144. DOI: 10.1109/ICDAR.2013.231

Mai, S.T., S. Goebl and C. Plant, 2012. A similarity model and segmentation algorithm for white matter fiber tracts. Proceedings of the IEEE 12th International Conference on Data Mining, Dec. 10-13, IEEE Xplore Press, Brussels, Belgium, pp: 1014-1019. DOI: 10.1109/ICDM.2012.95

Settles, B., 2009. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin, Madison.

Zhou, S., A. Zhou, J. Cao, W. Jin and Y. Fan *et al*., 2000. Combining sampling technique with DBSCAN Algorithm for clustering large spatial databases. Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp: 169-172. DOI: 10.1007/3-540-45571-X_20