International Journal For Technological Research In Engineering
Volume 3, Issue 8, April-2016 *ISSN (Online): 2347 - 4718*

CLUSTERING BASED ON DBSCAN

Ms. Samata.S.Huddar[1], Mr. Pramod[2], Mr. B R Prasad Babu[3]
[1]M Tech Research Scholar, [2]Associate Professor, [3]Professor and Head
CSE Dept, East Point College of Engineering and Technology, Bangalore, INDIA.

Abstract: DBSCAN is a density based clustering algorithm extracting the arbitrary shapes of the normal lanes from AIS data. This paper presents a parallel DBSCAN algorithm on top of Hadoop the experiment conducted in the paper shows that the proposed method can work well with maritime data although the performance is not satisfying. A discussion about the method's limitation and potential issues is shown at the end of the paper.
Key words: Maritime Surveillance; Clustering.

I. INTRODUCTION

Density based spatial clustering of applications with noise (DBSACN) is a data clustering algorithm proposed by martin Ester,Hans-peter Kriegel,Jorg Sander and Xiaowei Xu in 1996[1]. It is a density based clustering algorithm. Given a set of points in some space, it groups together points that are closely packed together (points with many nearby neighbors), making as outliers points that lie alone in low density regions (whose nearest neighbors are too far away). DBSCAN is one of the most common clustering algorithms and also most cited scientific literature. Numerous applications require the management of spatial data, i.e. data related to space. Spatial Database Systems(SDBS) (Gueting 1994)[2] are database systems for the management of spatial data. Increasingly large amounts of data are obtained from satellite images, X-ray crystallography or other automatic equipment. Therefore, automated knowledge discovery becomes more and more important in spatial databases. Several tasks of knowledge discovery in databases (KDD)[3] have been defined in the literature (Matheus, Chan&Piatetsky- Shapiro 1993)[4]. The task considered in this paper is class identification, i.e. the grouping of the objects of a database into meaningful subclasses. In an earth observation database, e.g., we might want to discover classes of houses along some river. Clustering algorithms are attractive for the task of class identification. However the e application to large spatial databases rises the following requirements for clustering algorithms

- Minimal requirements of domain knowledge to determine the input parameters, because appropriate values. These values often not knowing advance then dealing with large databases.
- Discovery of clusters with arbitrary shape, because the shape of clusters in spatial databases may be spherical, drawn-out, linear, elongated etc.
- Good efficiency on large databases, i.e. on databases of significantly more than just a few thousand objects. Clustering algorithms are attractive for the task of class identification in spatial databases.

However, the application to large spatial databases rises the following requirements or clustering algorithms: minimal requirements of domain Knowledge to determine the input parameters, discovery of clusters with arbitrary shape and coding efficiency on large databases. Thee well-known clustering algorithms offer no solution to the combination of these requirements in this paper, this resent the new clustering algorithm DBSCAN relaying on a density-based notion of clusters which is designed to discover clusters of arbitrary shape. DBSCAN only one input parameter and supports user in determining an appropriate value for it. In 2014, the algorithm was awarded the test of time award (an award given to algorithms which have received substantial attention in theory and practice) at the leading data mining conference, KDD[5]. The well-known clustering algorithms offer no solution to the combination of these requirements. In this paper, we present the new clustering algorithm DBSCAN. It requires only one input parameter and supports the user in determining an appropriate value for it. It discovers clusters of arbitrary shape. Finally, DBSCAN is efficient even for large spatial databases. The rest of the paper is organized as follows. We discuss clustering algorithms, evaluating them according to the above requirements. We present our notion of clusters which is based on the concept of density in the database. Even introduces the algorithm DBSCAN which discovers such clusters in a spatial database. We performed an experimental evaluation of the effectiveness and efficiency of DBSCAN using synthetic data and data of the SEQUOI2A0 00 benchmark[6]. At last concludes with a summary and some directions for future research.

II. ARCHITECTURE

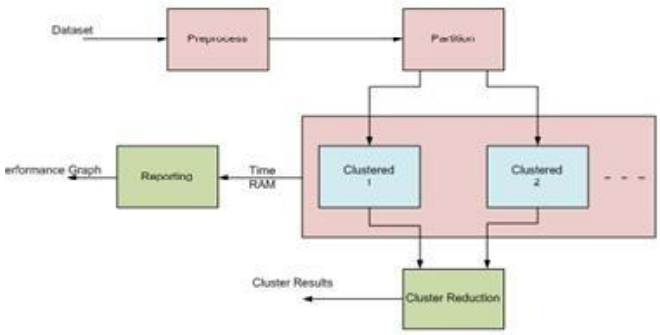

Figure 1: System Architecture.

DBSCAN Architecture explains how the data is accessed as shown in the Figure 1[7]. First of all dataset is given to find the particular data in the dataset. Dataset is input to the DBSCAN algorithm. These data set is processed and send to

partition the data, in the partition of data, dataset is partition into clusters, these clustering happens on the RAM, and time taken for the partition is reported by the performance Graph[8]. Then these clusters are reduced and gives the cluster results. And the particular data which is needed is accessed by the cluster result.

III. CLUSTERING ALGORITHM

Consider a set of points in some space to be clustered. For the purpose of DBSCAN clustering, the points are classified as core points, (density-) reachable points and outliers[9], as follows: A point p is core point if at least min Pts points are within distance d' of it and those points are said to be directly reachable from p. No points are directly reachable from a non- core point. A point q is reachable from p if there is a path p1,…pn with p1=p and p(n)=q, where each p(i+1) is directly reachable from pi(so all the points on the path must be core points, with the possible exception of q).All points not reachable from any other point are outliers.

Now if p is a core point, then it forms a cluster together with all points (core or non-core) that are reachable from it. Each cluster contains at least one core point. Non -core points can be part of cluster, but they form its "edge"[10], since they cannot be used to reach more points.

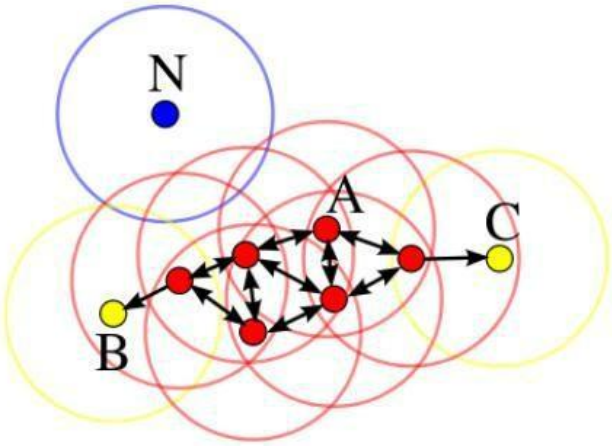

Figure 2: Clustered diagram.

In this diagram, min pts=3. Point A and the other red points are core points, because at least three Points surrounded it in and radius. Because they are all reachable from one another, they form a single cluster. Point B and C are not core points, but they are reachable from A (via other core points) and thus belong to the cluster as well. Point N is a noise point that is neither a core point nor density reachable. Reachability is not a symmetric relation since, by definition, no point may be reachable from a non-core point, regardless of distance (so a non-core point may be reachable, but nothing can be reached from it). Therefore a further notion of connectedness is needed to formally define the extent of the clusters found by DBSCAN. Two points' p and q are density –connected if there is a point 'O' such that both p and q are density reachable from 'O' density connectedness is symmetric. A cluster then satisfies two properties: All points within the cluster are mutually density-connected. If a point is density – reachable from any point of the cluster, it is part of the cluster as well.

IV. DBSCAN ALGORITHM.

DBSCAN requires two parameters: d (eps) and the minimum number of points required to form a dense region (minPts).It starts with an arbitrary staring point that has not been visited[11]. This point's d-neighborhood many points, a cluster is started. Otherwise, the point is labeled as noise. Note that this point might later be found in a sufficiently sized d- environment of a different point and hence be made part of cluster.

If a point is found to be dense part of a cluster, its d-neighborhood is also part of that cluster. Hence, all points that are found within the d- neighborhood are added, as is their own d-neighborhood when they are also dense. This process continues until the density- connected cluster is completely found. Then, a new unvisited point is retrieved and processed, leading to the discovery of a further cluster or noise.

As mentioned before, clustering algorithms are used to reduce data sets into groups, or clusters, which can be more readily analyzed and reasoned about. These clusters can then be used to make predictions about new data, or to find previously unnoticed connections among existing data. That makes clustering algorithms useful in a variety of fields, from machine learning, to data mining, to image analysis. For a given data set, there are multiple ways to find commonalities between the data in order to generate clusters. This characteristic of clustering has led to the development of several clustering algorithms. Each one of those algorithms uses different criteria to form clusters from the data. Among the most widely used algorithms for clustering we can find K-Means and DBSCAN.DBSCAN is a density-based clustering algorithm. Density based clustering algorithms define a cluster as an area that hasa higher data density than its surrounding area. In DBSCAN density is measured by analyzing whether a point has at least a minimum number of points (MinPts) inside a given radius (d). That is, if the _ neighborhood (points within d) of a given point has exceeded a particular density threshold (MinP ts) that given point, and its neighbors, form a cluster. The DBSCAN algorithm is described in pseudo code in Algorithm.

Algorithm 1 The DBSCAN algorithm

Input: A set of points X = {p1,p2,...,pn}, the distance threshold , and the minimum number of points required for a cluster MinPts.

Output: A set of labeled points X = {p1,p2,...,pn}, where each point has a flag corresponding to one of CORE, BORDER or NOISE and in the case of the flag being CORE or BORDER a corresponding cluster identifier.

clusterIdentifier ← next available cluster identifier
foreach unvisited point p ∈ X do

mark *p* as visited
N ← GETN $\text{EIGHBORS}(p, \epsilon)$
if |*N*| < *MinPts* then
p.flag ← NOISE
Else
p.clusterIdentifier ← *clusterIdentifier*
p.flag ← CORE
foreach $p^0 \in N$ do
if p^0 is not visited then
mark p^0 as visited
N^0 ← GETN $\text{EIGHBORS}(p', \epsilon)$
if $|N^0| \geq$ *MinPts* then
p^0.*flag* ← CORE
$N \leftarrow N \cup N^0$
Else
p^0.*flag* ← BORDER
end if
end if
if p^0 does not belong to any cluster then
p^0.*clusterIdentifier* ← *clusterIdentifier*
p^0.*flag* ← BORDER
end if
clusterIdentifier ← next cluster identifier
end for
end if
end for
Algorithm 1: DBSCAN ALGORITHM.

V. COMPLEXITY

DBSACAN visits each point of the database, possibly multiple times (e.g., as candidates to different clusters). For practical considerations, however, the time complexity is mostly governed by the number of region Query invocations. DBSCAN executes exactly one such query for each point, and if an indexing structure is used that executes a neighborhood query in O(log n)[12], an overall average runtime complexity of O(n log n) is obtained (if parameter d is chosen in a meaningful way, i.e such that on average only O(log n) points are returned). Without the use of an accelerating index structure, or on degenerated data (e.g. all points within a distance less than d), the worst case run time complexity remains O(n^2). The distance matrix of size (n^2-n)/2 can be materialized to avoid distance re computations, but this need O(n^2) memory, where as a non- matrix based implementation of DBSCAN only needs O(n) memory.

VI. CONCLUTION

DBSCAN does not require one to specify the number of clusters in the data priori, as opposed to k-means. DBSCAN can find arbitrary shaped clusters. It can even find a cluster completely surrounded by (but not connected to) a different cluster. Due to the MinPts parameter, the so- called single-link effect (different clusters being connected by a thin line of points) is reduced[13]. DBSCAN has a notation of noise, and is robust to outliers. DBSCAN requires just two parameters and is mostly insensitive to ordering of the points in the database. (However, points sitting on the edge of two different clusters might swap cluster membership if the ordering of the points is changed, and the cluster assignment is unique only up to isomorphism.) DBSCAN is designed for use with database can use that can accelerate region queries, e.g. using an R*tree[14]. The parameter minPts and d can be set by a domain expert, if the data is well understood. DBSCAN is not entirely Deterministic: border points that are reachable from more than one cluster can be part of either cluster, depending on the order the data is processed. Fortunately, this situation does not arise often, and has little impacted on the clustering result: both on core points and noise, DBSCAN is deterministic. DBSCAN is a variation that treats border points as noise, and this way achieves a fully deterministic result as well as a more consistent statistical interpretation of density- connected components[15]. If the data and scale are not well understood, choosing a meaningful distance threshold d can be difficult. Future research will have to consider the following issues. First, we have only considered point objects. Spatial databases, however, may also contain extended objects such as polygons. We have to develop a definition of the density in an Eps-neighborhood in polygon databases for generalizing DBSCAN. Second, applications of DBSCAN to high dimensional feature spaces should be investigated. In particular, the shape of the k-distance graph in such applications has to be explored.

REFERENCES

[1] Jain Anil K. 1988. Algorithms for Clustering Data. Prentice Hall. KaufmanL and RousseeuwR J. 1990. Finding Groups #~ Data: an Introduction to Cluster Analysis. John Wiley & Sons.
[2] Matheus C.J.; Chan P.K.; and Piatetsky-Shapiro G. 1993. Systems for Knowledge Discovery in Databases, 1EEE Transactions on Knowledge and Data Engineering 5(6):
[3] Wikipedia. (2015, April, 5) Scalability — Wikipedia, the freeencyclopedia. Accessed 22-July-2004. [Online]. Available: http://en. wikipedia.org/wiki/Scalability on DBSCAN
[4] K.-H. Lee, Y.-J. Lee, H. Choi, Y. D. Chung, and B. Moon, "Parallel dataprocessing with mapreduce: a survey," AcM sIGMoD Record, vol. 40,no. 4, pp. 11–20, 2012.
[5] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.
[6] Wikipedia. (2015, April, 4) Mapreduce — Wikipedia, the free encyclopedia. Accessed 22-July-2004. [Online]. Available: http://en. wikipedia.org/wiki/MapReduce
[7] B.-R. Dai and I.-C. Lin, "Efficient map/reduce-based dbscan algorithm with optimized data partition," in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*. IEEE, 2012, pp. 59–66.

[8] Y. He, H. Tan, W. Luo, H. Mao, D. Ma, S. Feng, and J. Fan, "Mrdbscan: an efficient parallel density-based clustering algorithm using mapreduce," in *Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on*. IEEE, 2011, pp. 473–480.

[9] Y. He, H. Tan, W. Luo, S. Feng, and J. Fan, "Mr-dbscan: a scalable mapreduce-based dbscan algorithm for heavily skewed data," Frontiers of Computer Science, vol. 8, no. 1, pp. 83–99, 2014.

[10] M. J. Berger and S. H. Bokhari, "A partitioning strategy for nonuniform problems on multiprocessors," Computers, IEEE Transactions on, vol. 100, no. 5, pp. 570–580, 1987.

[11] P. Wendell and M. Zaharai. (2015, February 13) Spark: A review of 2014 and looking ahead to 2015 priorities. [Online]. Available:https://databricks.com/blog/2015/02/13/spark-a-review-of-2014-and-looking-ahead-to-2015-priorities.html

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E.

[13] Duchesnay, "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011. (2015, April 12) Datasetloadingutilities. [Online]. Available: http://scikit-learn.org/stable/datasets/

[14] A. Guttman, R-trees: a dynamic index structure for spatial searching. ACM, 1984, vol. 14, no. 2. M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*. USENIX Association, 2012, pp. 2–2.

[15] Wikipedia. (2015, April, 5) Scalability— Wikipedia, the free encyclopedia. Accessed 22-July-2004. [Online]. Available: http://en. wikipedia.org/wiki/Scalability

