A Complexity Survey on Density based Spatial Clustering of Applications of Noise Clustering Algorithms

Boulchahoub Hassan[1], Rachiq Zineb[2], Labriji Amine[3], Labriji Elhoussine[4]
Laboratory of Systems Engineering (LaGeS), Hassania School of Public Works EHTP, Casablanca, Morocco[2]
Department of Mathematics and Computer Science, Faculty of Sciences Ben M'SIK, Casablanca, Morocco[1, 3, 4]

***Abstract*—Data Clustering is an interesting field of unsupervised learning that has been extensively used and discussed over several research papers and scientific studies. It handles several issues related to data analysis by grouping similar entities into the same set. Up to now, many algorithms were developed for clustering using several techniques including centroids, density and dendrograms approaches. We count nowadays more than 100 diverse algorithms and many enhancements for each algorithm. Therefore, data scientists still struggle to find the best clustering method to use among this diversity of techniques. In this paper we present a survey on DBSCAN algorithm and its enhancements with respect to time requirement. A significant comparison of DBSCAN versions is also illustrated in this paper to help data scientist make decisions about the best version of DBSCAN to use.**

Keywords*—Unsupervised learning; clustering; density clustering; DBSCAN

I. Introduction

The fast development of the internet and the availability of cheap mobiles, smart sensors and social networks applications allow users to generate a huge amount of data continuously. This rapid increase of data volume makes several domains difficult to be understood easily using only human capabilities. However many algorithms for clustering have been developed to guide data scientists to analyse and to understand data despite its volume. Nowadays, these algorithms play a crucial role in several sophisticated systems and applications including recommender systems, medical applications, face recognition, environmental assessment and anomalies detection [1][2][3][4][5]. To better understand any phenomena under investigation, clustering algorithms must extract correct and efficient statistics and trends, which is a very hard task, because results are often influenced by the nature of the real-world data which can be sparse, dense, spatial, high dimensional or even noisy. Therefore, algorithms must handle all complicated issues generated by data such as supporting volume increases, improving the scalability, processing high dimensional space, dealing with shaped structure and detecting outliers. The quality of clustering is also mainly influenced by the choice of the initial parameters such as number of clusters or the density radius. Thus, algorithms must vanish, optimize or even detect the parameters to use in order to detect meaningful clusters. To deal with all mentioned difficulties in real cases, many clustering approaches were raised including partitioning methods [6], hierarchical methods [7] and density based methods [8], etc.

In this paper, we are interested in density-based clustering, where clusters are defined by areas in which the density of the data points is high and clusters are separated from each other by areas of low density. We will focus especially on the DBSCAN algorithm [8] which can process spatial data efficiently and it can discard outliers properly. DBSCAN is a very simple and reliable technique, however it suffers from many limitations including its high complexity $O(n^2)$, its sensitivity to the local density variation, its dependence on initial parameters and its scalability failures. Therefore, it has undergone several improvements to make it efficient and to avoid its bastard chaos as effectively as possible. For instance, I-DBSCAN [9] and FDBSCAN [10] enhance the time requirement and minimize the deviation of results, MR-DBSCAN [11] improves scalability and deals with heavily skewed data and HDBSCAN [12] solves initial parameters issues, etc. We propose a comparison guide for all DBSCAN enhancements related to the complexity criterion and a repository for all DBSCAN versions related to time requirement is also presented in this work.

This paper is organized mainly in 4 sections: Sections 1 and 2 contains a brief refresh related to the clustering concept followed by an in-detail description about DBSCAN. Section 3 discusses the well-known DBSCAN improvements according to time complexity. Section 4 presents a comparison between DBSCAN versions based on time criteria.

II. Clustering Techniques

This section contains a brief description of partitional, hierarchical and density based clustering.

Clustering is the process of affecting each data object to a group based on the distance computation or on the similarity between each pair of observations. It is considered as the main process for many fields including image processing, pattern recognition, statistical data analysis and other business applications. Clustering methods can be broadly divided into several types including partitional, hierarchical, density based clustering, etc.

A. Partitional Clustering

Clustering has taken its roots from the partitioning method K-mean [6] which organizes all observations into an already

known number of groups (K). Each cluster is represented by its mean called centroid and objects are affected to the nearest cluster centroid. This method iterates many times over all observations to minimize the following objective function:

$$min \sum_{i=1}^{k} \sum_{x \in Si} ||x - \mu i||^2$$

Where Si is a Set of observations from a dataset, $Si \in \{S1, S2, \ldots, Sk\}$, k is the number of clusters and μi is the mean of points in Si.

K-mean is based on a very simple computation technique, however it is sensitive to outliers, data shapes and it assumes that clusters have roughly equal numbers of observations. In some cases, as mentioned in Fig. 1, it can lead to bad or even surprising results. Fig. 1(b), (f) and (d) are wrong clustering results.

The K-means method has relatively low time complexity and high computing efficiency, but it finds only compact and spherical shapes and it is still not suitable for non-convex data. Additionally, it needs prior knowledge about the number of clusters (K), it selects randomly the initial centroids. Thus, many improvements were done to overcome the limitations aforementioned such as the Partitioning Around Medoids (PAM) [13], Clustering Large Applications (CLARA) [14], and K-means for outlier detection [15]. Despite all efforts made, this type of clustering is not used when groups of data are expected to differ in size and shape, when the number of clusters is not known and when data contains noises. For instance, hierarchical and density clustering are explored to discover arbitrary shaped and meaningful clusters from large amounts of spatial data by preserving the spatial proximity of data objects.

B. Hierarchical Clustering

Hierarchical techniques seek to organize data objects into a tree structure representation called dendrogram. They are based on the computation of a symmetric distance matrix and they use some properly defined partitioning methods such as Ward's method [16], single or complete linkage. Several algorithms were invented under this type of clustering such as CURE [17] such as BIRCH [18].

As mentioned in Fig. 2, hierarchical techniques can be agglomerative or additive depending on where the algorithm starts processing the tree, from the top or from the bottom. Once the tree is built, hierarchical algorithms make splits in the additive processing or merge in agglomerative processing in order to find clusters. These cuts or merges decisions must be made properly thereby the quality of clusters will be better. For instance, as illustrated in Fig. 2, the level of cutting defines the number of clusters to detect. The first level gives rise to two clusters while the second one creates four clusters. Hierarchical algorithms are easy to understand and to implement. However, they rarely provide accurate results for mixed data types, they work poorly on very large data sets, they involve lots of arbitrary decisions and unfortunately, no adjustment can be performed once a merge or split decision has been executed. Many exceptions detected in partitioning and hierarchical clustering can be handled by using a density based clustering illustrated in the next section.

C. Density based Clustering

Density-Based Clustering refers to finding contiguous regions with high density among the dataset.

As mentioned in Fig. 3, these regions should be separated by low density regions called sparse regions. The idea behind such algorithms is that the clusters are represented by the detected dense regions and data objects in the sparse regions are typically considered noise/outliers.

In the next sections of this paper, we will focus on the most popular and the most cited density based algorithm (over 19430 times) called Density-Based Spatial Clustering of Applications with Noise [8].

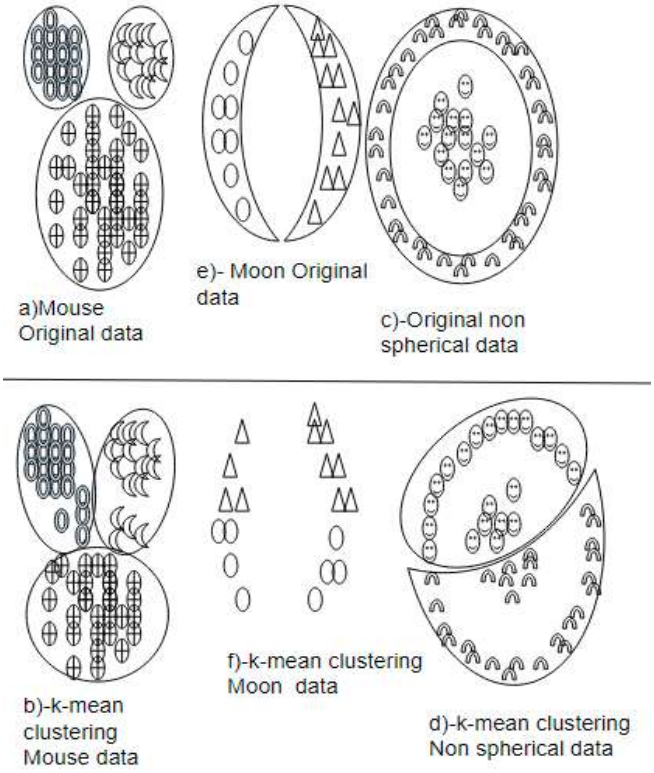

Fig. 1. K-Means Clustering Samples.

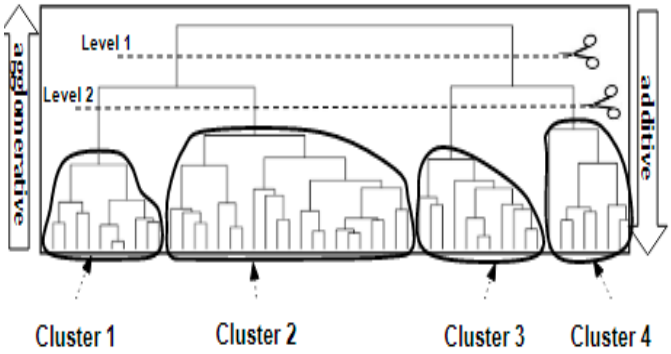

Fig. 2. Hierarchical Clustering Dendrogram.

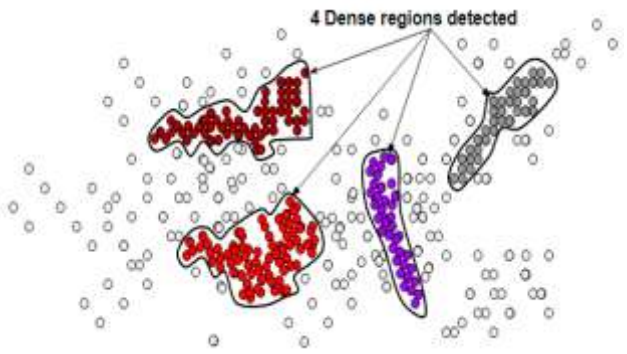

Fig. 3. Example for DBSCAN Clustering.

III. DBSCAN Basis

A. Definitions

DBSCAN received many scientific awards such as the test-of-time award from the leading data-mining conference KDD2014 for its good performance and its significant accuracy in clustering spatial data. The main purpose of DBSCAN is to detect arbitrary shaped clusters within a large data set and to effectively distinguish noises. It measure the density at any object O by counting the number of objects falling in a hyper sphere S(O,ε) where ε is a radius measured by an Euclidean distance. A region delimited by S(O,ε) is considered dense if the object O satisfies the following equation:

$$N\varepsilon\ (O) > MinPts\ where$$

$$N\varepsilon\ (O) = \{Q \in Dataset\ and\ distance\ (Q, O) < \varepsilon\}$$

$N\varepsilon\ (O)$ is the ε-neighbourhood of the object O and MinPts is the minimum number of points required to be present in the region to make hyper sphere S(O,ε) dense.

So, if objects share the same dense S(O,ε) then they belong to the same cluster. As mentioned before and to decide if a region is dense or sparse, this algorithm uses two parameters: an Euclidean distance threshold ε and a positive integer parameter MinPts.

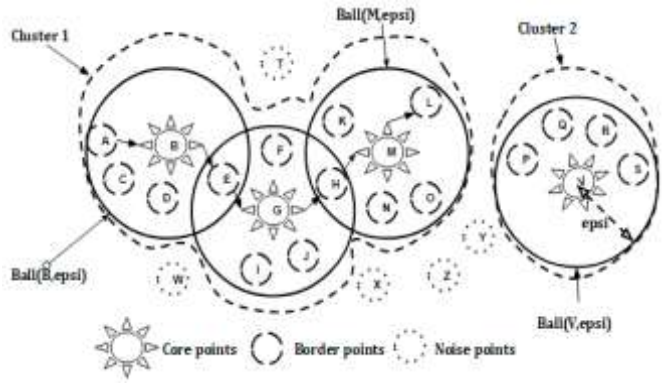

Fig. 4. .DBSCAN Core, Border and Noise Points.

As described below, DBSCAN introduces many definitions to categorize data objects into core, border or noisy objects [8].

DEFINITION 1: Core objects

An object O is considered core object if the number of objects inside the hyper sphere S(O,ε) is greater than MinPts parameter value. The points B, G, M are core objects in cluster 1 and V is a core object cluster 2.

DEFINITION 2: Border objects

An object is border if it belongs to some ε-neighbours of some core objects and the number of its own ε-neighbourhood is less than MinPts value. Thus, an object O is considered as a border object if it belongs to a cluster without being a core object. In Fig. 4, A,C,D,E,I,J,H,F,K,L,H,N and O are border objects for cluster 1 and P,Q,R and S are border objects for cluster 2.

DEFINITION 3: Noise objects

If an object O is not a core or border object then it is considered as a noise or outlier. In Fig. 4, the points T, W, X, Z, and Y are outliers.

DEFINITION 4: Directly Density reachability and Density reachability

If an object O is a core object, so all objects within the ε-neighbourhood of P are called directly density reachable objects from O. In Fig. 4, border objects A,C,D,E are directly reachable form the core point B and border objects P,Q,R,S are directly reachable from the core point V.

Two objects O1 and On are density reachable, if a chain of objects O1, O2, …, On is found within the dataset where Oi+1 is directly density-reachable from Oi with respect to the initial parameters ε, MinPts and $i\epsilon[1, n]$. For instance the chain A → B → E → G → H → M → L, in Fig. 4, makes a density link between the objects A and L. Thus A and L are density reachable.

DEFINITION 5: Maximality and Connectivity

Maximality: If a core object O belongs to a cluster, then all the objects density-reachable from O also belong to the same cluster.

Connectivity: If two objects O1, O2 belong to the same cluster so there is another object O in the same cluster such that both O1 and O2 are density-reachable from O. In Fig. 4, B and G are connected because they are density reachable through the chain B→E→G

B. DBSCAN Algorithm

Based on the previous definitions and the previously mentioned parameters ε and MinPts, we illustrate the DBSCAN algorithm in Table I.

TABLE I. DBSCAN ALGORITHM

DBSCAN ALGORITHM . DBSCAN(Data,ε,MinPts)

```
ClusterId = 0; // a cluster identifier
for each object O in Data do
   if O is not marked as "seen" then
      Mark O as "seen";
      Find Neighbors(ε,O,Data); // Neighbors of the object O using ε
param
      if  card(Neighbors(ε,O,Data))<MinPts then
          Mark O as "noise";
      else
          Mark O as "seen";
          ClusterId = ClusterId +1;
          Mark each object of  Neighbors(ε,O,Data) with cluster identifier
ClusterId;
          Add each object of Neighbors(ε,O,Data) which is not marked as
"seen" to the queue(ClusterId)
          while queue(ClusterId) is not empty do
               Take an object P from queue(ClusterId)  and mark it as
"seen"
               if card(Neighbors(ε,P,Data)>MinPts then
                   Mark each object of Neighbors(ε,P,Data) with cluster
identifier ClusterId;
                   if any object of Neighbors(ε,P,Data) is marked "noise"
then remove this mark.
                   Add each object of Neighbors(ε,P,Data) which is not
marked as "seen"  to
                   queue(ClusterId)
               end if
               Remove y from queue(ClusterId)
           end while
       end if
    end if
end for
Output all objects of Data along with their ClusterId or "noise" mark.
```

The previous algorithm describes DBSCAN where "Neighbors (ε, P, Data)" is the sub-set of objects in "Data" that are present in the hyper-sphere of radius at S(P, ε). "Card(Neighbors(ε,P,Data)" is the cardinality of the set "Neighbors(ε,P,Data)". Each object from "Data" is marked with a cluster identifier (ClusterId) which gives the cluster to which the object belongs or it is marked as "noise" indicating that the object is a noisy one. To distinguish between the objects which are processed from that which are not, the mark "seen" is used. Note that all objects of Neighbors(ε,P,Data) are initially marked as "noise", except the object P, then they can later become a border point of a cluster and hence the "noise" mark can be deleted.

According to the previous description, we can easily notice that DBSCAN does not require the pre-determination of the number of clusters and it requires only two parameters to determine when a region is considered to be dense or sparse, however it still suffers from several limitations including its high complexity which can reach $O(n\text{^}2)$, its failure with local density variation, its handicap related to data scalability and its huge memory consumption. Many works have been adopted to bring a significant optimization of the DBSCAN algorithm and to overcome its major drawbacks. For instance, E. Schubert et al. [19] discussed the relationship of the indexability of the dataset and the quality of clusters. They proposed some indicators of bad parameters to guide data scientists in choosing appropriate parameters ε and MinPts. For time reduction, B. Borah and D. K. Bhattacharyya used a sampling-based method [20] and J. Gan and Y. Tao used approximation techniques [21]. Derya Birant and Alp Kut tried to cover no spatial and spatial–temporal data by DBSCAN [22]. Moreover, other improvements were released to cluster in high dimensional space [23], to use parallel processing opportunities [24], [25] and to fix local density variation issues [26]. Knowing that complexity is a powerful criterion to decide about the efficiency of an algorithm, we propose a survey in the rest of this paper, a review of some well-cited DBSCAN extensions which significantly affect the time requirement.

IV. DBSCAN COMPLEXITY ENHANCEMENTS

Complexity criterion is among the ultimate indicators which qualify the efficiency of an algorithm. Thereby we decided to cover in this section some well-cited DBSCAN papers published between 2000 and 2019 and aiming to enhance the time requirement of the original algorithm. In the rest of this paper, "n" will represent the number of samples in the dataset and "d" will refer to the number of features studied. As mentioned in the first section, DBSCAN computes the empirical density for each dataset element and it measures mutual distances for the entire observations. Hence it requires a large volume of memory and a huge period of time to achieve large datasets clustering. Thus, it is qualified, by data scientists, as a very expensive algorithm and it is widely criticized due to its quadratic time requirement. Originally, Ester et al. [8] claimed that the DBSCAN will terminate in O (nlog(n)). However, the neighbourhood queries consume a big part of the running time. It requires $\sum_{p\in D}|N\varepsilon\,(p,D)| = O(n^2)$ to measure distances between all objects regardless of the initial parameters MinPts and ε. Fortunately, this time requirement can be reduced significantly to reach $O(n\,log_m n)$ [27] if some suitable indexing structure is used such as R*-tree [28] where m is the number of entries in a page of R*-tree. However, the use of R*-tree is suitable only when the dimensionality of the data is low. Thereby, researchers are still trying to run DBSCAN in some subquadratic time (i) by reducing the queries time and (ii) by minimizing the number of queries needed. As results of their efforts, many new methods appeared including hybrid methods [9] [29] which used only some accurate objects as prototypes rather than using all dataset objects. This approximation used by the hybrid methods can, in some cases, lead to clusters with bad quality. In the next paragraphs, we will weigh the pros and cons of some well cited DBSCAN time reducing methods.

B Borah et al. proposed IDBSCAN in 2004 to incorporate a sampling technique for searching the core object's neighbourhood. They used only outer objects as seeds and they ignored no representative objects. Therefore, they omit unnecessary queries by adding an extra function to the original algorithm based on Marked Boundary Objects (MBO) technique [20]. This function adds a complexity of O(sd) where s is the neighbourhood size. However, the overall complexity of IDBSCAN is $O(n\,log_m n)$ where m is the number of entries. Chen et al. [30] proposed an exact and approximate algorithm with $O(n^{(2-2/d+2)}\,polylog\,n)$ time for high dimensional space and $O(n^{1/2}\,polylog\,n)$ for two

dimensional spaces. P. Viswanath and Rajwala Pinkesh [9] proposed another fast hybrid density method called L-DBSCAN based on leaders clustering technique [31]. They derived some representative objects at the coarser level and others at the finer level of the clustering process. Authors used a first category of leaders to reduce the time requirement and second category to optimize the deviation of the results. This hybrid scheme uses only a set of pairs denoted by $\mathcal{L}^* = \{(\ell, \text{count}(\ell)) | \ell \in \mathcal{L}\}$ where ℓ is a leader and $\mathcal{L}$ a set of leaders. According to experiments showed by the authors, L-DBSCAN can run in $O((n+k)^2)$ where k represents the number of derived leaders which is much smaller than n. However, this technique can give raise to big margin error, when a leader is not originally dense but it is estimated dense according to $\mathcal{L}^*$. This method reduces the computation time, but it requires two additional thresholds: τc and τf.

P. Viswanath and V. Suresh Babu [29] enhanced the density approximation of leaders in their technique called rough-DBSCAN by combining the leaders clustering method [31] and the rough set approach [32]. They added a mapping between every leader and its belonging objects (followers). This mapping is represented by $\mathcal{L}^*$ = {(followers (l), count (l)} where l is a leader form $\mathcal{L}$. Then, they used a lower and upper approximation, as shown in equation 1, 2 and 3, to find the exact neighbours of a leaderN_ε (l, D).

$$N_{\varepsilon,\text{lower}-\tau}(l, D) \subseteq N_\varepsilon(l, D) \subseteq N_{\varepsilon,\text{upper}+\tau}(l, D) \quad (1)$$

$$N_{\varepsilon,\text{lower}-\tau}(l, D) = \cup_{l \in \mathcal{L}_1} \text{followers}(l) \quad (2)$$

$$N_{\varepsilon,\text{upper}+\tau}(l, D) = \cup_{l \in \underline{\mathcal{L}_1}} \text{followers}(l) \quad (3)$$

where $\mathcal{L}_1 = \{l_i \in \mathcal{L} \mid ||l_i - l|| < \varepsilon - \tau$

and $\underline{\mathcal{L}_1} = \{l_i \in \mathcal{L} \mid ||l_i - l|| \leq \varepsilon + \tau\}$

Rough-DBSCAN needs only $O(n+k^2)$ where k is the number of leaders, but it improves the clustering quality by minimizing the approximation error.

FDBSCAN [33] is another non-linear searching algorithm proposed by B. Liu, in 2007, to reduce redundant searching by using a fast merging algorithm. It sorts objects using dimensional coordinates and then selects only unlabelled objects outside a core object's neighbourhood in order to decrease region queries. Another interesting paper is proposed, in the same year, by Yi-Pu Wu et al. [34] to optimize the process of Nearest Neighbour Search by using Locality-Sensitive Hashing (LSH) technique. Authors used the hash collisions to detect and represent similarities between two objects A and B form a dataset D. On the other hand, to capture object similarity they compute the probability distributions over a set of hash functions $\mathcal{H}$ as $Pr_{h \in \mathcal{H}}[h(A) = h(B)] = S(A, B)$ where $h \in \mathfrak{H}$ and S is a similarity function defined as $S: DxD \rightarrow [0,1]$. This LSH algorithm makes a significant decrease in DBSCAN running time which becomes O (N) and maintains the quality of detected clusters [35].

Cheng-fa Tsa and Chien-Tsung Wu, inspired by the fast merging method of FDBSCAN [33], proposed the GF-DBSCAN algorithm to segment data into several grid-cells and to limit the neighbourhood searches only to the cell scope instead of exploring the entire grid. They merge clusters if they are intersected and the overlapping objects include some core object. By introducing this grid approach and this merging process, GF-DBSCAN minimizes significantly the number of searches and increases the clustering accuracy [36]. Gunawan [27] demonstrated that DBSCAN's performance can be improved to$O(n \log n)$ by applying the following process in order (i) partitioning data using a grid-cell (ii) determining all core points (iii) merging density-connected core points into clusters and finally (iiii) determining border points and noise. He used the hash table to discard cells without any point. However this faster algorithm is experimented only in two dimensional space. Therefore, J.Gan and Y.Tao extend Gunawan's thesis to $\mathbb{R}^d$and they get a running time of $\{O((nlogn)^{\frac{4}{3}}) if\ d = 3\}$ and $\{O(n^{\left(2-\frac{2}{\lceil \frac{d}{2} \rceil + 1} + \delta\right)}) if\ d >= 4\}$ where the parameter δ specifies the accuracy of the approximation. J.Gan and Y.Tao also proposed a new algorithm called ρ-approximate suitable for large datasets which can be computed in an expected time of $O\left(\frac{n}{\delta^{d-1}}\right)$. They are inspired by Chen et al.'s paper [30] which already discussed how to compute DBSCAN in $O(n \log n + n/\delta^{d-1})$.

GPUs opportunities and parallelization strategies are also used by some algorithms including G-DBSCAN algorithm [37] to speed up the original algorithm. G-DBSCAN constructs firstly a data graph G(O, E) where O are objects (nodes) connected by edges E if they are within a minimum proximity R (threshold parameter) of each other. Then it identifies clusters by using breadth-first search (BFS) technique [38]. Thereby, a complexity of O (n + ne) is added by the BFS search where 'ne' is the number of edges. G-DBSCAN uses graphics processing units (GPU's) capabilities to achieve acceleration greater than 100×, but unfortunately it doesn't reduce the original complexity.

The DBSCAN neighbour search operation can be optimized by using a graph-based index structure method, as demonstrated by K. Mahesh Kumar, and A. Rama Mohan Reddy [39]. Their idea is to prune out outliers objects early to vanish unnecessary distance computations which may be introduced by noises. RNN-DBSCAN [40] uses reverse nearest neighbour counts and k nearest neighbour graph traversals to estimate observation density. It reduces complexity of DBSCAN by using a single parameter (choice of k nearest neighbours) and also improves the ability of handling large variations in cluster density (heterogeneous density). Mark de Berg et al. [41] presented another O (n log n) approximate algorithm for DBSCAN in two dimensional space. They represented data objects using a smaller box graph $\mathcal{G}\_box$ where nodes are disjoint rectangular boxes with a diameter of at most ε and edges connect pairs of boxes within distance ε from each other. Then they detected another graph $\mathcal{G}\_core$ including only core points where the connected components of $\mathcal{G}\_core$ are considered as clusters. Mark de Berg et al. improved the quality of clusters by assigning borders to their nearest core point rather than the first cluster that finds them. Table II summarizes all aforementioned DBSCAN complexity enhancements.

TABLE II. TIME COMPLEXITIES OF THE WELL CITED DBSCAN VERSIONS

Year	Name	Time	Method used	Ref.
1996	DBSCAN	$O(n\hat{}2)$	Density based technique	[8]
		$O(n\log n)$	R*-tree	
2000	FDBSCAN: A fast DBSCAN algorithm	$O(n\log n)$	Representative points technique	[10]
2004	IDBSCAN	$O(n\ log_m n)$	Marked Boundary Objects (MBO)	[20]
2005	GEOMETRIC ALGORITHMS FOR DENSITY-BASED DATA CLUSTERING	$O\left(n^{2-\frac{2}{d+2}}\right)$ if d>=3 $O(n^{1.5}\ polylog\ n)$ for d=2	Computational geometry techniques. ε-fuzzy distance	[30]
2006	L-DBSCAN	$O((n+k)^2)$ K is the number of leaders	Leaders Clustering Method [31]	[9]
2006	FDBSCAN	$O(n\log n)$	Representative points technique kernel function	[33]
2007	A Linear DBSCAN Algorithm Based On LSH	$O(n)$	LSH : Locality sensitive hashing	[35]
2009	Rough DBSCAN	$O(n+k^2)$ K is the number of leaders	Set theory [32] leaders clustering method[31]	[29]
2009	GF-DBSCAN	1/100 of the time cost of FDBSCAN	Fast merging and Grid cells.	[36]
2010	TI-DBSCAN	Up to three orders of magnitude faster than DBSCAN.	Triangle Inequality property	[42]
2013	A faster algorithm for DBSCAN	$O(n\log n)$ for d=2	Grid partition Hash Table	[27]
2013	G-DBSCAN	Over than 100x faster than its sequential version using CPU	Data-graph (node and edges) breadth-first search BFS [38]	[37]
2015	ρ-approximate DBSCAN	$O((nlogn)^{\frac{4}{3}})$for d=3 $O(n^{2-\frac{2}{\lceil\frac{d}{2}\rceil+1}+\delta})$ for d>3 δ is a small positive constant	Approximation technique	[21]
2016	A fast DBSCAN clustering algorithm by accelerating neighbour searching using Groups method	improves the speed of DBSCAN by a factor of about 1.5–2.2	Graph-based method	[39]
2017	Faster DBSCAN and HDBCAN in Low-Dimensional Euclidean Spaces	$O(n\log n)$ $time\ for\ \mathbb{R}^2$	Box graph method	[41]

V. CONCLUSION

DBSCAN is a powerful technique for data clustering; however it still suffers from its huge time requirement which can reach $O(n^2)$ in the worst case. This paper weighs the pros and cons of the well-known and well-cited DBSCAN variations with respect to the time requirement. We present the current state of art related to DBSCAN complexity and also we mentioned some techniques used to enhance the original version of the algorithm. According to the papers studied, researchers can use the leaders clustering method, Graph-based method, breadth-first search BFS, Triangle Inequality Property or Locality sensitive hashing to bring new enhancements in this field. We noticed that DBSCAN complexity vary between $O(\mathrm{n})$ and $O(n^2)$. Another analysis of all these DBSCAN variations based on real data experiments will be presented in our further works.

ACKNOWLEDGMENT

Authors would like to express their special thanks of gratitude to Mr Labriji and Mr Rachik for their able guidance and support in completing this manuscript. We would also like to extend our gratitude to the anonymous reviewers whose thoughtful comments and suggestions will lead to improving this manuscript.

REFERENCES

[1] H. S. Emadi and S. M. Mazinani, "A Novel Anomaly Detection Algorithm Using DBSCAN and SVM in Wireless Sensor Networks," Wireless Personal Communications, vol. 98, no. 2. pp. 2025–2035, 2018, doi: 10.1007/s11277-017-4961-1.

[2] M. K. Najafabadi, M. N. Mahrin, S. Chuprat, and H. M. Sarkan, "Improving the accuracy of collaborative filtering recommendations using clustering and association rules mining on implicit data," Computers in Human Behavior, vol. 67. pp. 113–128, 2017, doi: 10.1016/j.chb.2016.11.010.

[3] S. Khanmohammadi, N. Adibeig, and S. Shane Bandy, "An improved overlapping k-means clustering method for medical applications," Expert Systems with Applications, vol. 67. pp. 12–18, 2017, doi: 10.1016/j.eswa.2016.09.025.

[4] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2015, doi: 10.1109/cvpr.2015.7298682.

[5] F. Yin and C.-L. Liu, "Handwritten Chinese text line segmentation by clustering with distance metric learning," Pattern Recognition, vol. 42, no. 12. pp. 3146–3157, 2009, doi: 10.1016/j.patcog.2008.12.013.

[6] J. B. MacQueen, "ON THE ASYMPTOTIC BEHAVIOR OF K-MEANS." 1965, doi: 10.21236/ad0629518.

[7] R. Sibson, "SLINK: An optimally efficient algorithm for the single-link cluster method," The Computer Journal, vol. 16, no. 1. pp. 30–34, 1973, doi: 10.1093/comjnl/16.1.30.

[8] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," 1996, Accessed: Dec. 08, 2020. [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.121.9220.

[9] P. Viswanath and R. Pinkesh, "l-DBSCAN : A Fast Hybrid Density Based Clustering Method," 18th International Conference on Pattern Recognition (ICPR 06). 2006, doi: 10.1109/icpr.2006.741.

[10] Z. Shui, "FDBSCAN: A Fast DBSCAN Algorithm," 2000, Accessed: Dec. 15, 2020. [Online]. Available: https://www.semanticscholar.org/paper/FDBSCAN%3A-A-Fast-DBSCAN-Algorithm-Shui/d6d1e7e468035b63138d6a4de4ca5685fb700808.

[11] 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, doi: 10.1007/s11704-013-3158-3.

[12] R. J. G. B. Campello, Ricardo J G, D. Moulavi, and J. Sander, "Density-Based Clustering Based on Hierarchical Density Estimates," Advances in Knowledge Discovery and Data Mining. pp. 160–172, 2013, doi: 10.1007/978-3-642-37456-2_14.

[13] "Kaufman, L. and Rousseeuw, P.J. (1990) Partitioning around Medoids (Program PAM). In Kaufman, L. and Rousseeuw, P.J., Eds., Finding Groups in Data An Introduction to Cluster Analysis, John Wiley & Sons, Inc., Hoboken, 68-125. - References - Scientific Research Publishing." https://www.scirp.org/(S(czeh2tfqyw2orz553k1w0r45))/reference/ReferencesPapers.aspx?ReferenceID=1771062 (accessed Dec. 04, 2020).

[14] L. Kaufman and P. Rousseeuw, "Clustering Large Applications (Program CLARA)," 2008, doi: 10.1002/9780470316801.CH3.

[15] S. Chawla and A. Gionis, "k-means-: A Unified Approach to Clustering and Outlier Detection," 2013, Accessed: Jan. 06, 2021. [Online]. Available: https://pdfs.semanticscholar.org/70f4/5be50599f12a1b682a192c3c48ebda0bb1c4.pdf.

[16] J. H. Ward, "Hierarchical Grouping to Optimize an Objective Function," Journal of the American Statistical Association, vol. 58, no. 301. pp. 236–244, 1963, doi: 10.1080/01621459.1963.10500845.

[17] S. Guha, R. Rastogi, and K. Shim, "Cure: an efficient clustering algorithm for large databases," Information Systems, vol. 26, no. 1. pp. 35–58, 2001, doi: 10.1016/s0306-4379(01)00008-4.

[18] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: an efficient data clustering method for very large databases," 1996, Accessed: Jan. 03, 2021. [Online]. Available: http://dl.acm.org/citation.cfm?id=233324.

[19] E. Schubert, J. Sander, M. Ester, H.-P. Kriegel, and X. Xu, "DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN," ACM transactions on database systems, vol. 42, no. 3, p. 6, 2017, Accessed: Dec. 15, 2020. [Online].

[20] B. Borah and D. K. Bhattacharyya, "An improved sampling-based DBSCAN for large spatial databases," International Conference on Intelligent Sensing and Information Processing, 2004. Proceedings of. doi: 10.1109/icisip.2004.1287631.

[21] J. Gan and Y. Tao, "DBSCAN Revisited: Mis-Claim, Un-Fixability, and Approximation," 2015, Accessed: Dec. 17, 2020. [Online]. Available: http://dl.acm.org/citation.cfm?id=2737792.

[22] "ST-DBSCAN: An algorithm for clustering spatial–temporal data," Data Knowl. Eng., vol. 60, no. 1, pp. 208–221, Jan. 2007, Accessed: Dec. 08, 2020. [Online].

[23] L. Ertöz, M. Steinbach, and V. Kumar, "Finding Clusters of Different Sizes, Shapes, and Densities in Noisy, High Dimensional Data," Proceedings of the 2003 SIAM International Conference on Data Mining. 2003, doi: 10.1137/1.9781611972733.5.

[24] G. Liu, B. Qiu, and L. Wenyin, "Automatic Detection of Phishing Target from Phishing Webpage," 2010 20th International Conference on Pattern Recognition. 2010, doi: 10.1109/icpr.2010.1010.

[25] Y. He et al., "MR-DBSCAN: An Efficient Parallel Density-Based Clustering Algorithm Using MapReduce," 2011 IEEE 17th International Conference on Parallel and Distributed Systems. 2011, doi: 10.1109/icpads.2011.83.

[26] P. Liu, D. Zhou, and N. Wu, "VDBSCAN: Varied Density Based Spatial Clustering of Applications with Noise," 2007 International Conference on Service Systems and Service Management. 2007, doi: 10.1109/icsssm.2007.4280175.

[27] A. Gunawan, "A faster algorithm for DBSCAN," 2013, Accessed: Dec. 28, 2020. [Online]. Available: https://www.semanticscholar.org/paper/A-faster-algorithm-for-DBSCAN-Gunawan/138f3e2aac21ca81fb7bf093ebae07859111e6dd.

[28] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The R*-tree: an efficient and robust access method for points and rectangles," Proceedings of the 1990 ACM SIGMOD international conference on Management of data - SIGMOD '90. 1990, doi: 10.1145/93597.98741.

[29] P. Viswanath and V. Suresh Babu, "Rough-DBSCAN: A fast hybrid density based clustering method for large data sets," Pattern Recognition Letters, vol. 30, no. 16. pp. 1477–1488, 2009, doi: 10.1016/j.patrec.2009.08.008.

[30] D. Z. Chen, M. Smid, and B. Xu, "GEOMETRIC ALGORITHMS FOR DENSITY-BASED DATA CLUSTERING," International Journal of Computational Geometry & Applications, vol. 15, no. 03. pp. 239–260, 2005, doi: 10.1142/s0218195905001683.

[31] H. Spaeth, "Cluster analysis algorithms for data reduction and classification of objects," 1980, Accessed: Jan. 08, 2021. [Online]. Available: https://cds.cern.ch/record/102044.

[32] Z. Pawlak, "Rough sets," International Journal of Computer & Information Sciences, vol. 11, no. 5, pp. 341–356, Oct. 1982, Accessed: Dec. 29, 2020. [Online].

[33] B. Liu, "A Fast Density-Based Clustering Algorithm for Large Databases," 2006 International Conference on Machine Learning and Cybernetics. 2006, doi: 10.1109/icmlc.2006.258531.

[34] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," 2004, Accessed: Jan. 12, 2021. [Online]. Available: http://dl.acm.org/citation.cfm?id=997857.

[35] Y.-P. Wu, J.-J. Guo, and X.-J. Zhang, "A Linear DBSCAN Algorithm Based on LSH," 2007 International Conference on Machine Learning and Cybernetics. 2007, doi: 10.1109/icmlc.2007.4370588.

[36] C.-F. Tsai and C.-T. Wu, "GF-DBSCAN: a new efficient and effective data clustering technique for large databases," 2009, Accessed: Dec. 15, 2020. [Online]. Available: https://www.semanticscholar.org/paper/GF-DBSCAN%3A-a-new-efficient-and-effective-data-for-Tsai-Wu/909e93cbf1867e2b5af089810bdbb8352e75ff53.

[37] G. Andrade, G. Ramos, D. Madeira, R. Sachetto, R. Ferreira, and L. Rocha, "G-DBSCAN: A GPU Accelerated Algorithm for Density-based Clustering," Procedia Computer Science, vol. 18. pp. 369–378, 2013, doi: 10.1016/j.procs.2013.05.200.

[38] E. F. Moore, The Shortest Path Through a Maze. 1959.

[39] K. M. Kumar, K. Mahesh Kumar, and A. Rama Mohan Reddy, "A fast DBSCAN clustering algorithm by accelerating neighbor searching using Groups method," Pattern Recognition, vol. 58. pp. 39–48, 2016, doi: 10.1016/j.patcog.2016.03.008.

[40] A. Bryant and K. Cios, "RNN-DBSCAN: A Density-Based Clustering Algorithm Using Reverse Nearest Neighbor Density Estimates," IEEE Transactions on Knowledge and Data Engineering, vol. 30, no. 6. pp. 1109–1121, 2018, doi: 10.1109/tkde.2017.2787640.

[41] M. de Berg, A. Gunawan, and M. Roeloffzen, "Faster DB-scan and HDB-scan in Low-Dimensional Euclidean Spaces," Feb. 28, 2017.

[42] M. Kryszkiewicz and P. Lasek, "TI-DBSCAN: Clustering with DBSCAN by Means of the Triangle Inequality," in Rough Sets and Current Trends in Computing, Jun. 2010, pp. 60–69, Accessed: Jan. 01, 2021. [Online].