Received April 27, 2021, accepted June 5, 2021, date of publication June 14, 2021, date of current version June 24, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3089036

Comparative Analysis Review of Pioneering DBSCAN and Successive Density-Based Clustering Algorithms

ADIL ABDU BUSHRA AND GANGMAN YI
Department of Multimedia Engineering, Dongguk University, Seoul 04620, South Korea

Corresponding author: Gangman Yi (gangman@dongguk.edu)

This work was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) through the Ministry of Education under Grant NRF-2019R1F1A1064019.

ABSTRACT The density-based spatial clustering of applications with noise (DBSCAN) is regarded as a pioneering algorithm of the density-based clustering technique. It provides the ability to handle outlier objects, detect clusters of different shapes, and disregard the need for prior knowledge about existing clusters in a dataset. These features along with its simplistic approach helped it become widely applicable in many areas of science. However, for all its accolades, the DBSCAN still has limitations in terms of performance, its ability to detect clusters of varying densities, and its dependence on user input parameters. Multiple DBSCAN-inspired algorithms have been subsequently proposed to alleviate these and more problems of the algorithm. In this paper, the implementation, features, strengths, and drawbacks of the DBSCAN are thoroughly examined. The successive algorithms proposed to provide improvement on the original DBSCAN are classified based on their motivations and are discussed. Experimental tests were conducted to understand and compare the changes presented by a C++ implementation of these algorithms along with the original DBSCAN algorithm. Finally, the analytical evaluation is presented based on the results found.

INDEX TERMS Unsupervised learning, clustering, DBSCAN, spatial database.

I. INTRODUCTION

Grouping of data objects is a necessary task in a wide range of studies such as medical diagnosis (clustering algorithms in identifying cancerous data [1]), civil engineering [2], academics [3], biology [4], [5], and networking (clustering algorithms in wireless sensor network-based applications [6], [7]). This process can be a supervised or unsupervised classification based on the information provided by a dataset. Supervised classification identifies data objects where known labels or classes for the different data objects are presented [8]–[10]. It is termed supervised because each data object is already labeled in a particular group and the classification process is simply the trivial task of assigning the object to the appropriate class. When no labels are assigned for data objects, however, grouping is performed through unsupervised classification [8], [11], [12]. The process of grouping a given dataset into classes in unsupervised classification is called "clustering" [13]–[17].

Clustering is the process of partitioning the data into a groups that are similar as possible given a set of data objects. [17], [18]. For continuous data, a distance-based approach can be applied to determine the similarity between data objects. Distance functions such as Euclidean distance [13], [19], cosine distance [13], [20], or the Pearson correlation measure [21]–[23] can serve as a similarity measure between pairs of data objects. Closer data objects are more similar and likely to be grouped in the same class compared with objects that are farther apart. Through this mechanism, clustering algorithms are used as tools to partition a given dataset into objects classes or "clusters" that have more in common with each other than with others.

Clustering techniques have been applied in marketing and sales [23], [24] for businesses to group their customers with similar purchases and traits in order to run a targeted marketing strategy. They have also been employed in spam filtering [25] to scrape through sections of an email and detect

The associate editor coordinating the review of this manuscript and approving it for publication was Liangxiu Han.

features associated with spams, in identifying fake news [26] from the media based on the coverage content, and several other tasks. Such real world implementations of clustering are helpful to identify all kinds of unknown patterns present in data.

Many clustering algorithms have been developed with their own interpretations of grouping data objects into different classes. These algorithms employ different yet sometimes overlapping tactics in their implementations. Therefore, it is difficult to make a distinctive classification of categories for these algorithms [14], [27], [28]. For the sake of providing an overview of the clustering strategies, however, it is generally accepted to distinguish them into five methods [13], [29]: partitioning, hierarchical, grid-based, model-based and density-based methods, as illustrated on Fig. 1.

Given a dataset of n objects, partitioning methods [29] work by constructing k partitions of the dataset, where each partition represents a cluster and $k \leq n$. Each of these partitioned groups is ensured to have at least one object as a member, with each data object belonging to exactly one group (except in some cases of fuzzy partitioning methods [30], [31]). The k-means clustering algorithm [24], [32], [33] is a popular heuristic method representing each cluster by the mean value of the objects in the cluster called a "centroid." The inclusion of a particular data object into a cluster is determined by which cluster's centroid it is most closest to. The k-medoids algorithm [33] is a similar algorithm, with each cluster being represented by an object located near the center of the cluster.

Hierarchical methods [33] meanwhile regard the decomposition of a given dataset into a hierarchy of objects as the crux of their operation. These methods can be "agglomerative", meaning they apply a bottom-up approach to form a hierarchy. Agglomerative approaches initiates with each data object set as a cluster; and proceed to merge these clusters iteratively until a termination condition is satisfied. Conversely, the "divisive" or top-down approach works by first initiating the whole dataset as a single cluster and iteratively splitting the cluster into smaller clusters instead. Chameleon [34] and BIRCH [35] are examples of hierarchical clustering.

Grid-based methods [18], [36] focus on the dataspace by forming a grid structure. Clustering operations are then performed on each cell of the grid structure instead of the data objects themselves. This quantization of the dataspace helps the efficiency of algorithms, such as STING [37] and WaveCluster [38] become independent of the number of data objects but dependent on the cells of the grid structure.

Model-based clustering [18], [39] techniques work by presuming that the given data were generated by some model, then finding the model that best fits the dataset. The clusters are then detected according to the model along with the cluster labels for the data objects.

The fifth distinguished method and the focus of this article along with its pioneering algorithm density-based spatial clustering on applications with noise (DBSCAN) [40], [41], is the density-based clustering technique [18], [42].

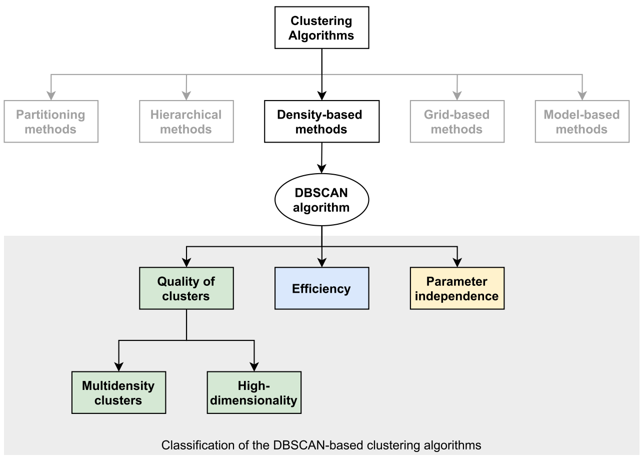

FIGURE 1. An overview of the several clustering strategies. Generally, a clustering algorithm falls under one of the five methods. Further classification for the successive clustering algorithms inspired by the pioneering DBSCAN algorithm is introduced.

The formal definitions of the clustering model and the DBSCAN algorithm were first introduced together in 1996, on the Knowledge Discovery in Databases (KDD) data mining conference publication. The core idea behind the density-based clustering method assumes that a cluster is a region in a dataspace with a high density of data objects. This clustering model brought unique features different from the earlier mentioned implementations. It introduced the ability to form clusters of irregular shapes, detect outliers in the dataspace, and identify clusters without prior knowledge of the classes present in the dataset. It is a practical algorithm, and the DBSCAN, has been applied in several fields of study such as civil engineering [43], chemistry [44], spectroscopy [45], social sciences [46], [47], medical diagnostics [48], remote sensing [49], [50] computer vision [51], automatic identification systems (AIS) [52], [53] and anomaly detection [54], [55]. Its successful implementation on real-world applications has led it to receive the Special Interest Group on KDD test-of-time award [41].

The DBSCAN has served as a stepping-stone to several other clustering algorithms that aim to improve on the original algorithm and its limitations. This paper examines the definitions introduced by the density-based clustering method and how the original DBSCAN procedure works, identifying the challenges DBSCAN faces and discussing the subsequent density based clustering algorithms inspired by it. Specifically, the contributions of this paper are as follows.

1. It provides a comprehensive review of density-based clustering mechanisms. In terms of the pioneering algorithm DBSCAN, this paper introduces several implementations in the literature that aim to reduce its limitations. This paper can serve as a guide for future researchers focusing on the density-based clustering algorithms.

2. It provides a study of the DBSCAN-inspired implementations by classifying them under three general categories. These categories are derived from the major limitations of the

original algorithm. Hence, successive algorithms can generally be identified under one of these introduced categories.

3. An empirical analysis of six density-based clustering algorithms provides a quantitative comparison between different implementations. Synthetic datasets are used to compare the accuracy of the cluster outputs, and several real-world datasets detail the computational time required for each dataset to complete.

Accordingly, the rest of this paper is organized as follows. Section II investigates the DBSCAN algorithm and its implementation, features, and shortcomings. Section III is dedicated to the various DBSCAN-based clustering algorithms. Based on the motivation of each algorithm, Section III discusses the algorithms under three distinct classifications. In Section IV, the experimental results for six clustering algorithms are reported and evaluated. Finally, the Section V concludes the paper by providing and overview of the study.

II. THE DBSCAN ALGORITHM

The DBSCAN [40] algorithm interprets clustering based on the notion that regions with a high density of data objects that are separated by sparser regions in a dataspace are identified as clusters. In addition, DBSCAN uses two input parameters ε and *minPts*, to identify the density estimation of a region surrounding a particular data object. The algorithm uses these two parameters to estimate the density of a particular data object's local region. Moreover, ε refers to the radius of an object's local region (neighborhood), whereas *minPts* is the minimum number of data objects required within that radius in order to be a cluster. Therefore, for a given dataset X, the DBSCAN algorithm calculates the local density of an object $x_i, x_i \in X$ as the total number of objects in its ε-neighborhood (i.e., cardinality of $N_\varepsilon(x_i)$), where

$$N_\varepsilon(x_i) = \{x_j \in X : \forall j,\ dist(x_i, x_j) < \varepsilon\}. \tag{1}$$

Through (1), DBSCAN can find arbitrarily shaped clusters. Next, by considering the number of objects in a neighborhood, three different types of objects appear. *Core objects* are those where their ε-neighborhood contains at least *minPts* number of objects (i.e., $|N_\varepsilon(x_i)| \geq minPts$, for some $x_i \in X$). These objects indicate that, for the given values of ε and *minPts*, their neighborhood objects (along with the core objects themselves) form a cluster. The number of objects in *border objects'* ε-neighborhood is less than the given *minPts*; however, for a border object x_j, a core object x_i exists where $x_j \in N_\varepsilon(x_i)$. The third type is *noise objects*. These objects do not have at least *minPts* number of objects in their ε-neighborhood and are not members of any core object's neighborhood. Beyond this, the DBSCAN algorithm sets further definitions regarding the reachability between pairs of data objects. The definitions are presented as follows.

A. DIRECTLY DENSITY REACHABILITY

An object $x_i \in X$ is directly density reachable from an object $x_j \in X$ for a given ε and *minPts* if: (a) x_i is a member of the ε-neighborhood of x_j, and (b) x_j is a core object:

$$DirReach(x_i, x_j) \Leftrightarrow x_i \in N_\varepsilon(x_j) \wedge |N_\varepsilon(x_j)| \geq minPts. \tag{2}$$

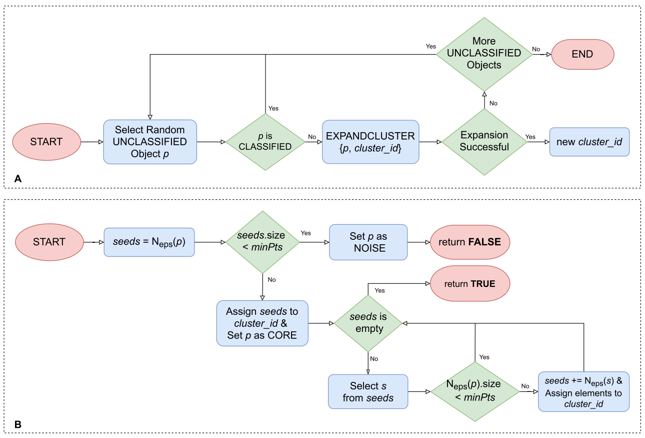

FIGURE 2. **Flowchart illustration of the original density-based spatial clustering of applications with noise (DBSCAN) algorithm. The EXPANDCLUSTER operation (B) can be subdivided from the core process. Based on the flowchart of DBSCAN (A), we can determine the algorithm operates in a recursive process. The EXPANDCLUSTER module returns a *true* value if a specific data object successfully builds clusters around its region; otherwise, it returns *false*.**

B. DENSITY REACHABILITY

An object $x_i \in X$ is density reachable from an object $x_j \in X$ with respect to a given ε and *minPts* if a series of objects $o_1, \ldots, o_n$, exists, where $o_1 = x_j$ and $o_n = x_i$ such that o_{i+1} is directly-density reachable from o_i:

$$\begin{aligned} &Reach(x_i, x_j) \\ &\quad \Leftrightarrow \exists o_i, \ldots, o_n \in X : o_1 = x_j \wedge o_n = x_i \\ &\quad \wedge\ \forall i \in \{1 \ldots n-1\} : DirReach(o_i, o_{i+1}). \end{aligned} \tag{3}$$

By this definition, two border objects that appear in the same cluster might not be density reachable from each other due to the core-object constraints set for the series of objects $o_1, \ldots, o_n$. However, they are directly density reachable to a common core object. Density connectivity defines this property.

C. DENSITY CONNECTIVITY

An object $x_i \in X$ is density-connected to an object $x_j \in X$ with respect to a given ε and *minPts* if there is another object o such that both x_i and x_j are density reachable from o with respect to ε and *minPts*:

$$Connect(x_i, x_j) \Leftrightarrow \exists o \in X : Reach(o, x_i) \wedge Reach(o, x_j). \tag{4}$$

Based on these notions, DBSCAN defines a cluster as a maximal set of density-connected objects concerning density reachability. That is, a cluster C with respect to ε and *minPts* is a nonempty subset of X such that:

a. $\forall x_i, x_j$: if $x_i \in C$ and x_j is density-reachable from x_i with respect to ε and *minPts*, then $x_j \in C$, and

b. $\forall x_i, x_j \in C$: x_i is density-connected to x_j with respect to ε and *minPts*

Finally, noise objects are a set of objects in the dataset that do not belong to any of its clusters.

The DBSCAN algorithm employs a two-step mechanism to determine the set of objects that satisfies the definitions provided in the datasets. These sets of maximal objects in terms of their density reachability are labeled into their respective clusters. The algorithm starts by choosing a random unclassified core-object from the dataset and considers it as "seed." It then collects all the density-reachable objects from the seed to build a cluster that contains the seed object. This two-step process is repeated until no more unclassified objects remain.

The collection of density-reachable objects from a seed is performed through the procedure EXPANDCLUSTER. It takes in a core object as a seed, finds all the objects within its ε-neighborhood by performing REGIONQUERY, and for each identified and unclassified object, places it in the same cluster as the seed. For all the core-objects found during REGIONQUERY, their ε-neighborhood objects are also collected and placed in the same cluster.

The overview of the DBSCAN algorithm on Fig. 2 reveals that it is a recursive process. For each object it considers to be a seed, extracting its ε-neighborhood objects through REGIONQUERY requires traversing through the entire dataset, at most. This alone yields a time complexity of $O(n)$ for a region query of a single object. When considering all the REGIONQUERY executions performed for each of the seeds throughout the run of the algorithm, the overall complexity of the algorithm rises to $O(n^2)$. This complexity could be reduced using a spatial index structure, such as the R*-tree, which requires $O(\log n)$ in order to perform region queries. Such an approach means the DBSCAN algorithm can find clusters in a dataset at $O(n \log n)$.

The original DBSCAN algorithm was successful in identifying clusters with arbitrary shapes. This feature is appreciated, especially in spatial databases where clusters can be spherical or even straight, bending, and other shapes. The characteristics of DBSCAN to identify regions with high density that are separated by sparser spaces allows its clusters' shapes to be determined by the dataset under examination as opposed to other clustering methods, such as the partitional clustering k-means, which always assumes a spherical shape for its clusters. In addition, DBSCAN can identify isolated data objects and is able to assign them as noise.

Partitional clustering methods, in comparison, divide the dataspace into an assigned k different clusters, without considering the effects of noise objects on the identified clusters. The DBSCAN also requires minimal knowledge to determine the input parameters, as the number of clusters that are generated is not directly dependent on the user but the size of the local regions of objects. This dependence contrasts with algorithms such as k-means clustering, where it takes the number of clusters to be generated as an input parameter.

However, the DBSCAN has some notable limitations. The algorithm is heavily dependent on the user-specified ε parameter. Based on the value set for ε, the clusters found by the

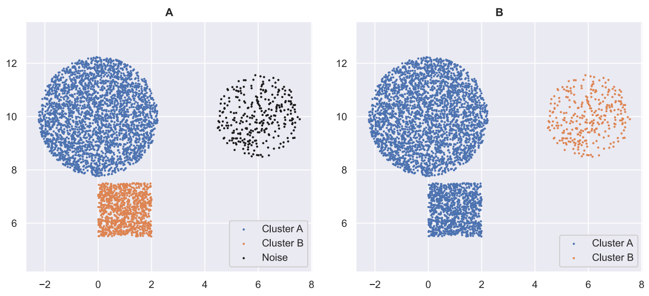

FIGURE 3. How groups of objects with different densities could hinder the quality of DBSCAN's results. (A) Cluster results generated when a low ε parameter value is chosen, causing clusters of sparser density to be falsely regarded as noise. (B) A higher value of ε produces a possible merging of two separate clusters. In either case, the DBSCAN algorithm did not manage to identify all three separate clusters correctly.

algorithm could vary widely. As Fig. 3 illustrates, a higher value of ε tends to inadvertently group a set of data objects into a single cluster that should otherwise have been clustered separately. A lower value might misclassify a set of data objects that belong in a cluster as noise if the density in the cluster's region does not satisfy the preset value of ε. Intuitively, we can also observe that treating ε as a global density parameter hinders the algorithm's ability to identify clusters with varying local densities. Many real-world datasets are characterized by data objects that are grouped together in different density ranges. When DBSCAN is applied on such a dataset with ε as the global parameter, either multiple clusters join into one (a large value of ε), or groups of data objects with a local density less than ε are assigned as noise (a small value of ε).

Furthermore, the EXPANDCLUSTER procedure within the algorithm is worth examining for its efficiency. As discussed, this procedure collects all the density-reachable objects from a particular core object to produce a cluster. This mechanism requires calling the REGIONQUERY operation on each of the ε-neighborhood elements of the core object. This requirement translates into a substantial portion of the algorithm's computational complexity, as a single REGIONQUERY by itself works by iterating across all the data objects in the dataset.

Several clustering algorithms have since been produced that recognize these limitations and propose new methods to alleviate them. In the next section, these various DBSCAN-based clustering algorithms are discussed.

III. VARIOUS DBSCAN-BASED ALGORITHMS

Many clustering methods have been introduced based on the notion of density proposed in the original DBSCAN publication. These methods were inspired by the DBSCAN and aim to reduce its limitations. In this section, we study the proposed properties of some of these methods and identify how they work to improve on the DBSCAN algorithm. To provide a clearer overview of these DBSCAN-based methods, we group these algorithms under three different classifications, each focusing on a major limitation of the

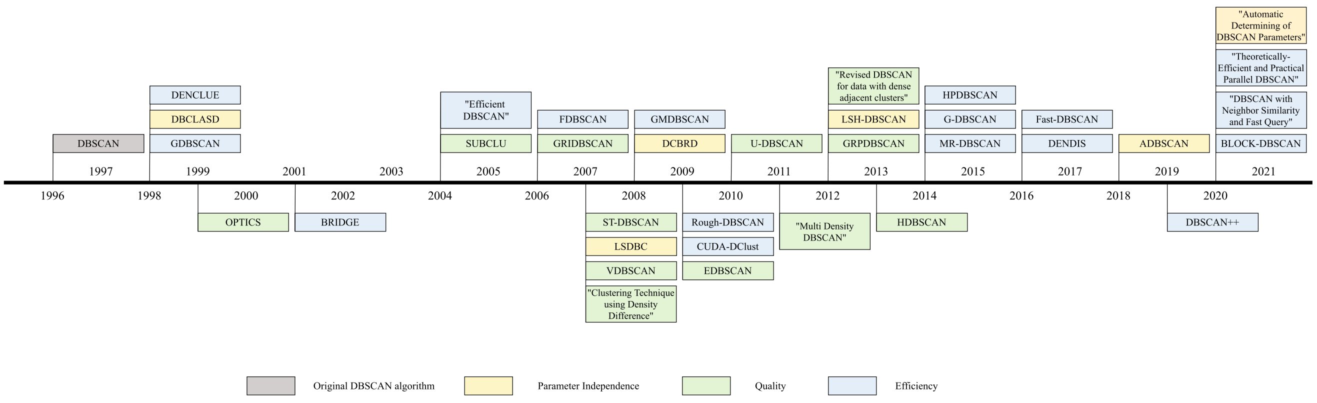

FIGURE 4. Timeline of the various DBSCAN-based algorithms. Each algorithm has been grouped under one of three categories: "Parameter Independence," "Quality of Clusters," or "Efficiency" based on improvements made over the original DBSCAN algorithm.

original DBSCAN. Fig. 4 plots these clustering algorithms on a timeline. Each of the algorithms' implementations is also described briefly in Table 1.

A. QUALITY OF CLUSTERS

The original algorithm is especially adept at handling spatial datasets that exhibit continuous data objects of irregular shapes. This feature, coupled with its ability to adequately identify anomalies and its automatic detection of cluster amounts within a dataset, is evidence of its superiority to other clustering mechanisms. However, in terms of the quality of clusters it produces, the DBSCAN faces two major challenges: datasets that exhibit multidensity structures in their data space and high dimensionality. As both of these are definitive properties of real-world databases, the original DBSCAN required the reimplementation of its method to address these two hindrances. We discuss the various DBSCAN-based algorithms that aim to solve these problems.

1) MULTIDENSITY DATASETS

One of the areas where DBSCAN faces a challenge is discovering clusters of varied densities in a dataset. As ε serves as a global input parameter, a set of objects grouped in a region with a density estimation greater than the ε value is regarded as noise. Several methods have been proposed to solve this problem and provide an analytical study to demonstrate their approaches.

Ordering points to identify the clustering structure (OPTICS) [57] was proposed to produce an ordering of a dataset representing its density-based clustering structure that corresponds to a wide range of parameter values. While OPTICS does not explicitly assign a cluster to each of the data objects, it instead holds the information necessary for an extended DBSCAN to assign cluster relationships to all the data objects. This cluster ordering is represented graphically using a reachability plot. Additionally, OPTICS introduced the notions of core distance and reachability distance to plot such a graph.

a: CORE DISTANCE

For an object $x_i \in X$, with respect to ε and $minPts$, its core distance is the smallest distance $\varepsilon^{`}$ possible to its $\varepsilon$-neighborhood $x_j \in X$ such that $x_i$ would be a core object with respect to $\varepsilon^{`}$ if $x_j \in N_\varepsilon(x_i)$. Otherwise x_i's core distance is undefined:

$$CoreDist(x_i) = \begin{cases} undefined, & \text{if } |N_\varepsilon(x_j)| < minPts \\ minPtsDist(x_i), & \text{otherwise.} \end{cases} \quad (5)$$

b: REACHABILITY DISTANCE

For an object $x_i \in X$, with respect to ε and $minPts$, its reachability-distance to another object $x_j \in X$ is the smallest distance such that x_i is directly density-reachable from x_j, if x_j is a core-object:

$$ReachDist(x_i) = \begin{cases} undefined, & \text{if } |N_\varepsilon(x_j)| < minPts \\ \max(CoreDist(x_j), & dist(x_j, x_i)). \end{cases} \quad (6)$$

Based on this, the results of the OPTICS algorithm generate an ordering of objects in the dataset corresponding to the objects' reachability distances. The reachability plot graphically depicts this where the reachability-distance values of each object are plotted.

The advantage of this clustering method is the relative insensitivity of the reachability plot to the input parameter ε. In the reachability plot, the effect of ε is on the number of clustering levels that are output on the graph. Smaller values of ε produces more objects with undefined reachability distances, disregarding clusters with lower densities. The optimal value for ε is the smallest possible value of ε such that the density-based clustering for ε and $minPts$ generates only one cluster with almost all the points of the dataset as members. Such a value would yields all the available clustering levels in the reachability plot.

The density-based clustering based on hierarchical density estimates (HDBSCAN) [64] introduces a hierarchical clustering method that proposes an improvement of OPTICS. It builds on the preexisting notions of OPTICS by applying

TABLE 1. Features of the DBSCAN-based algorithms.

Categories	Algorithms	Time Complexity [*]	Features
Quality of Clusters	GDBSCAN [56]	$O(n^2)$	A generalized DBSCAN algorithm intended to cluster objects according to both their spatial and nonspatial attributes.
	OPTICS [57]	$O(n^2)$	Produces an ordering of a dataset that represents its density-based cluster structure, which corresponds to a broad range of parameter settings.
	SUBCLU [58]	$O(2^d n^2)$	Uses the monotonous property of density-connectivity and efficiently prunes subspaces to cluster high-dimensional datasets.
	ST-DBSCAN [59]	$O(n^2)$	Proposed extensions for discovering core objects and adjacent clusters for the ability to discover data objects based on their spatial and temporal values.
	VDBSCAN [60]	$O(n^2)$	Methods applied for the extraction of various values of ε based on the k-dist plot in order to identify clusters with varying densities.
	EDBSCAN [61]	-	Keeps track of local density variations within a cluster by calculating the density variance of a core object relative to its ε-neighborhood. Proposed to find varied density clusters.
	Multidensity DBSCAN [62]	-	An extended DBSCAN algorithm that replaces the parameter ε with "local cluster density", to find clusters with differing densities.
	GRPDBSCAN [63]	-	Grid-based DBSCAN algorithm with referential parameters uses grid partition techniques to improve efficiency, whereas the auto-generated ε detects robust clusters.
	Revised DBSCAN algorithm to cluster data with dense adjacent clusters [50]	$O(n^2)$	Identifies an issue in DBSCAN where border objects of adjacent clusters are not always grouped to the same cluster and provides an improved algorithm for datasets with dense structures.
	HDBSCAN [64]	$O(dn^2)$, d: number of dimensions	An accelerated hierarchical density-based clustering to support variable density clusters, which removes the dependency on ε and the concept of border objects.
Efficiency	DENCLUE [65]	$O(n)$	Focus on clustering multimedia databases with high volume of noise objects. Identification of clusters by determining density-attractors. This also allows for mathematical description of arbitrarily shaped clusters.
	FDBSCAN [66]	$o(n^2)$	Proposed to decrease the frequency of region queries executed by selecting ordered and unclassified data objects outside a core object's neighborhood as seeds in order to expand clusters
	GMDBSCAN [67]	Linear to n, provided dimension is not high	Grid-based approach to divide the dataspace into partitions and applying spatial index to improve the efficiency. This technique also allows its to set local $minPts$ parameter to find multi-dense clusters
	Rough-DBSCAN [68]	$O(n)$	A hybrid clustering technique that first applies the leaders clustering method to derive prototypes that produce density-based clusters. Claims a complexity of $O(n)$.
	CUDA-DClust [69]	$O(n^2)$	GPU-based implementation of density-based clustering. Provides parallel computation as well as a novel index structure for use in GPUs.
	MR-DBSCAN [70]	$O(n^2)$	Implementation of the MapReduce parallel programming platform. A four-stage paradigm is adopted.
	HPDBSCAN [71]	-	A "highly parallel", scalable DBSCAN employing OpenMPI/MPI [72] to break the sequential process of the original algorithm and accelerate the neighborhood searches in a distributed environment.
	Fast-DBSCAN [73]	$O(n + nd)$, d: maximum number of REGIONQUERY	A novel graph-based index structure to accelerate the region query operation that is also scalable for high dimensional dataset.
	DBSCAN++ [74]	$O(mn)$, m: number of sampled objects	Reduces the frequency of regional query operation by defining a chosen subset of objects to run on. Selects the subsets of an object by using the greedy initialization method and approximates k-center objects
	BLOCK-DBSCAN [75]	Average complexity of $O(n)$	An approximate clustering approach to process large scale data by using a cover tree to compute densities of unclassified data objects. Introduces a fast approximate algorithm to identify core blocks that are density reachable.
	DBSCAN with neighbor similarity and fast nearest neighbor query [76]	$O(n(\frac{1}{seeds}logn + minPts^2))$	Aimed to remove the redundant region queries in the original DBSCAN through triangle inequality, neighbor similarity, and fast neighbor search algorithm. Uses a cover tree in its implementation.
	Theoretically-efficient and practical parallel DBSCAN [77]	Exact-DBSCAN: $O(n \log n)$ Approximate-DBSCAN: $O(n)$	Proposes a parallel exact 2D DBSCAN using a grid construction method for partitioning objects, a high-dimensional exact DBSCAN, and an approximate DBSCAN designed with quadtree construction and querying.
Parameter Independence	DBCLASD [78]	-	Distribution-based clustering method. Proposes the notion of nearest neighbor distances and computes their probability distribution to define a cluster. This notion is independent of any input parameters
	LSDBC [79]	$O(n^2)$	Aimed to automate clustering by using the local density information. Local maxima of density are discovered using k-nearest neighbor density estimation
	DCBRD [80]	$O(nk +^2 k + nm)$ k: number of circular regions $m = \frac{n}{k}$	Partitions the data into overlapped circular or hyper spherical regions and uses the best region to generate the neighborhood of a given data object. This lifts the requirement of any input parameter.
	LSH-DBSCAN [81]	$O(kn)$, k: dimension of index structure	A clustering approach that implements a k-dimensional tree structure to treat the time complexity of the DBSCAN algorithm and also remove its dependence on input parameters through an automatic k-distance graph computation
	ADBSCAN [82]	-	Adaptive DBSCAN was proposed to discover clusters of varying densities in the dataset by automating the value of ε. The incremental approach is implemented in order to detect clusters at particular values of ε.
	Automatic determining of the DBSCAN parameters [83]	-	A method to extract the ε and $minPts$ value for a given dataset. Uses the k-dist function to correctly determine the sharp increases ("knee") of the ordered k-nearest neighbor distances as ε.

[*] Time complexity of the algorithms retrieved from the original published articles. Papers that did not discuss their methods' complexity are left blank (-). The reported complexities do not assume the use of index structures, unless specified.

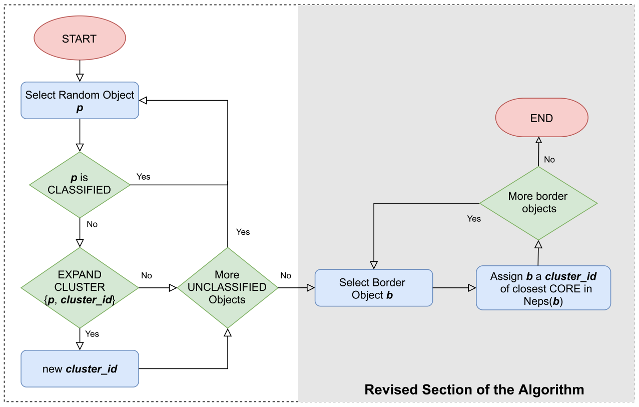
START
Select Random Object *p*
p is CLASSIFIED
Yes
No
EXPAND CLUSTER {*p*, *cluster_id*}
No
More UNCLASSIFIED Objects
Yes
No
Yes
new *cluster_id*
Select Border Object *b*
Assign *b* a *cluster_id* of closest CORE in Neps(*b*)
More border objects
Yes
No
END
Revised Section of the Algorithm

FIGURE 5. Flowchart illustration of the revised DBSCAN algorithm. The revised section of the algorithm as detailed in the shaded region of the illustration, ensures that border-objects are always assigned to the clusters closest to them; as opposed to the original DBSCAN, which assigns borders to the first cluster instance in which they appear. This revised method guarantees consistent cluster formations in all runs of the algorithms.

the mutual reachability graph. The mutual reachability graph is a complete graph, where objects on X are the vertices, and the weight of each edge is the mutual reachability distance between the respective pairs of objects. Based on the definitions of the DBSCAN algorithm, clusters in the graph are the connected components of the ε-core objects, with the remaining objects regarded as noise.

The varied density-based spatial clustering of applications with noise (VDBSCAN) [60] is another method that adopts the plotting mechanism of the objects in the dataset in order to extract clusters with differing densities. Before operating the DBSCAN algorithm, VDBSCAN proposes to compute the distance of the objects in the dataset to their k^{th} nearest neighbor ($k - dist$). The objects are sorted by their $k - dist$ values and plotted on a graph. The $k-dist$ plot produces sharp changes at suitable values of ε. The DBSCAN algorithm is computed for each corresponding values of ε. At each run of the algorithm, objects detected in a cluster at ε_i, are marked as clustered at density level i. Marked objects do not participate in further computation of DBSCAN. The two-step process of identifying varied-density clusters based on $k - dist$ is independent of ε as an input parameter but instead requires the specification of the k^{th} nearest neighbor in order to draw the plot.

Another limitation the DBSCAN algorithm might face is during the clustering of adjacent dense regions in the dataspace. As the algorithm starts by choosing an arbitrary unlabeled object, its subsequent operation of finding neighborhood core objects and performing expansions could result in different outputs of cluster members where border objects of separate groups of objects are closer together.

The revised DBSCAN algorithm to cluster data with dense adjacent clusters (revised DBSCAN) [50] describes this matter in depth by demonstrating how multiple iterations of the DBSCAN algorithm could produce different cluster structures for the same dataset. The proposed method focuses on the idea of density-reachable chains of objects to solve this problem. From the definition of density reachability, border objects could only be present in the density-reachable sets of objects as the last element of the chain. Hence, border objects do not participate in the expansion of clusters. Therefore, a new concept is proposed, the core-density-reachable objects, where the chains of objects are $x_1, \ldots, x_n$ for $x_i \in X$ and $|N_\varepsilon(x_i)| \geq minPts, i \leq n$ (all the objects in the chain are core-objects). The EXPANDCLUSTER is then further revised in order to identify border objects for the corresponding expanded clusters. Due to the implementation of core-density-reachability, the border objects would remain unclassified during the EXPANDCLUSTER operation. Therefore, for each border object, the closest core object is identified and clustered to the core-density-reachable chain in which the core object belongs to. Fig. 5 illustrates this concept in a flowchart visualization of the revised DBSCAN algorithm.

2) HIGH-DIMENSIONAL DATASETS

The original DBSCAN, and several other traditional clustering algorithms often fail to find meaningful clusters in datasets characterized by high dimensional and consequently sparse data space. This challenge has been termed as the "curse of dimensionality," where high dimensional datasets contain meaningful clusters deep in the subsets of their feature space, yet such algorithms as the DBSCAN, which considers the whole dimensional space while clustering, fail to identify them. This challenge is present due to DBSCAN's use of distance measures between data objects to determine similarity. In high dimensional datasets however, dimensions with irrelevant information would hide meaningful clusters in the dataspace with noisy data and ultimately hinder DBSCAN's attempt to identify clusters.

Multiple methods in the literature have proposed dimensionality reduction to find such meaningful clusters. These methods work by removing features deemed "irrelevant" or "redundant" in order to filter the dimensions that can be clustered appropriately. Other methods based on attribute transformation have also been proposed, such as implementing the principal component analysis (PCA) [84] to generate functions of attributes. However, such methods of mapping the whole feature space into lower spaces have major challenges of their own. The lack of intuitive meaning of the generated attributes is one challenge, whereas such methods as PCA do not return the desired results [85].

Hence, subspace clustering techniques have been proposed to overcome these problems. Subspace clustering approaches work by automatically detecting clusters in the multiple subspaces of the original data space. Besides overcoming the mentioned problems of the previous implementations, subspace clustering is not limited to detecting clusters in a single subspace. Therefore, information on objects clustered differently in varying subspaces is reserved.

One approach that implements the subspace clustering techniques for density-based clustering is the *density-connected subspace clustering (SUBCLU)* [58] algorithm. During the clustering of high dimensional datasets, the DBSCAN algorithm tries to detect clusters by considering all the dimensions for each pair of objects and applying the necessary distance measures. However, SUBCLU works to generate clusters that are formed within dimension subsets present in the dataset. The algorithm uses the concept of density-based clustering introduced by DBSCAN as its underlying clustering method. Provided we have a $d-$dimensional dataset X, and $A = a_1, \ldots, a_d$ as the set of all attributes in X; a subspace is any subset $S \subseteq A$. SUBCLU automatically identifies these subsets of dimensions (subspaces) in which clusters exist. In addition, SUBCLU applies the notion of monotonicity of density-connected sets to locate these subspaces. Given two objects that are density connected in some subspace $S \subseteq A$, the notion indicates that these two objects are also density-connected in any subspace $T \subseteq S$:

$$\begin{aligned} &Connect^{S}(x_i, x_j) \\ &\quad \Rightarrow Connect^{T}(x_i, x_j), \quad \forall T \subseteq S \subseteq A, \{x_i, x_j\} \in X. \end{aligned} \tag{7}$$

Additionally, SUBCLU uses this monotonicity property of density-connected sets to prune subspaces that need not be investigated, (i.e., if at least a single subspace $T \subset S$ fails to have a density-connected set, then subspace S does not have a set of density-connected objects either).

B. EFFICIENCY

The process of identifying all the density reachable elements from a particular core object involves the iterative calls of REGIONQUERY for all the ε-neighbors of a core-object. As the REGIONQUERY is a costly singular operation by itself, improved ideas have been proposed in terms of either minimizing the number of calls made to the REGIONQUERY, or optimizing the process of the operation itself by proposing novel index structures that facilitate the complexity of neighbor search operations.

The fast density-based clustering algorithm for large databases (FDBSCAN) [66] was proposed to address the complexity brought by the original DBSCAN's recursive execution of REGIONQUERY. The FDBSCAN introduced a method to decrease the frequency of calls to this operation by identifying the redundancy in DBSCAN's regional queries during its expansion phase. When EXPANDCLUSTER is called on an object $x_i \in X$, it finds the core-objects within the neighborhood of x_i and performs REGIONQUERY on each core object. However, as the neighbor objects in the core objects' ε-neighborhood intersect, repetition of REGIONQUERY is bound to occur, which increases the complexity of the operation.

Hence, FDBSCAN proposes selecting some representative objects within a region to solve this. When selecting seed objects to expand on and form a clusters, FDBSCAN chooses orderly unlabeled objects outside a core object's neighborhood. The ordering is performed by sorting the objects using a certain dimensional coordinate. Two clusters formed using this method can later be merged together if a core object exists within the intersection of the two cluster regions. This approach ensures that REGIONQUERY does not repeat on a single object that is in the ε-neighborhood of several core objects.

The *Rough-DBSCAN* [68] algorithm is a hybrid clustering algorithm proposed to decrease the time complexity of the original DBSCAN algorithm. The algorithm first searches for model objects within the dataset called leaders by using the leaders clustering method. Afterward, these objects are used to produce clusters through density-based clustering. This mechanism enables Rough-DBSCAN to compute the clusters in $O(n)$, which is an improvement on the DBSCAN algorithm. This computation is possible due to Rough-DBSCAN's implementation of first deriving k leader objects containing density information using the leader clustering method, which takes $O(n)$ and later applies the DBSCAN on these k leader objects only.

The leader method [86] is a partitioning clustering method. Its advantage is in terms of efficiency, which is linear with the number of objects in a given dataset. The procedure runs as follows. Given a threshold distance τ, it maintains a set of leader objects L that initializes to empty and builds iteratively. For each object $x_i \in X$; if a leader object $l \in L$ exists so that the distance between x_i and l is not greater than the threshold distance τ, then x_i is assigned to the cluster of l and assigned as the follower of object l. An object is assigned to only a single leaders object. Through this scheme, each object in L can be regarded as a representative for the cluster of objects that follow it. However, two problems are present with the leader method.

1. Although we can guarantee that two follower objects within the same group have a distance of at most 2τ, there is no way to ensure that followers within different groups have a distance of at least τ.

2. There can be multiple l values with distances of less than or equal to τ to and object $x_i \in X$. Thus, repeated runs of the algorithm on the same dataset could provide different clustering results.

To alleviate these issues, Rough-DBSCAN provides a modified implementation of the leaders clustering algorithm. The modification offers an ordering of the leader objects as they are derived, which allows the algorithm to choose the closest leader object l for an element x_i.

After generating the leaders objects with their corresponding set of followers, the proposed algorithm Rough-DBSCAN moves on to its density-based clustering mechanism. The algorithm uses ε and *minPts* parameters to partition the set of leaders derived from the dataset, which is later expanded into the whole dataset by replacing each of the leaders objects with their set of followers. This hybrid implementation of the leaders clustering algorithm on top of the DBSCAN algorithm helps in gaining efficiency by reducing the iterations of the DBSCAN from being linear with the number of objects in the dataset to k amount, which corresponds to the number of leaders produced.

A grid and density-based fast spatial clustering algorithm (GMDBSCAN) [67] is another hybrid clustering algorithm that joins the concepts of grid-based clustering along with an underlying DBSCAN implementation to produce an efficient clustering mechanism. The GMDBSCAN exploits the efficiency provided by grid-based applications that partition the whole dataspace into a hypersphere in order to limit the necessary traversal of the REGIONQUERY operation of the DBSCAN to only a limited data objects. As the algorithm obtains a grid-structure within the whole dataspace, REGIONQUERY operations are performed within a specified grid structure each time. Thus, the processing speed of the algorithm is independent of the number of data objects within the dataset.

The DBSCAN algorithm works by performing distance computations between each pairs of data objects in the dataset. However, this is unnecessary because the computation of the distance between two data objects with a distance greater than the threshold of parameter ε is useless for the process of forming density-based clusters. In addition, GMDBSCAN aims to remove such redundant operations by limiting the search space of a single data object. Grid structures are formed on the dataspace, forming hypercubes of grid blocks with their width set to the value of the distance threshold ε. This structure guarantees that, during the REGIONQUERY sub-operation for an object $x_i \in X$, its neighbor objects would only appear within the adjacent grids of where x_i is, (i.e., for a single data object in the dataset, an object $x_i \in X$ only has to perform distance computations with objects within its adjacent grids instead of against each object within the dataspace).

C. PARAMETER INDEPENDENCE

The dependence of DBSCAN on its input parameters ε and *minPts* to define its density and provide a density estimation means there requires a degree of knowledge about the dataset to determine the best values for the inputs. This requirement is aggravated by the fact that varying ε values could return highly different clustering results. Adaptive methods have been proposed to alleviate this that automatically compute for the optimal ε values to perform the DBSCAN algorithm.

Density clustering based on the radius of data (DCBRD) [80] is a density-based clustering algorithm that requires no input parameters. It performs its clustering of data objects with DBSCAN as its core method. The algorithm focuses on partitioning the data into overlapped circular regions (hyperspheres for multidimensions) and uses the best region to select the neighborhood of a data object. The proposed algorithm divides the clustering process into two stages. First, through some distance measure, the algorithm creates several overlapping circles (hyperspheres). Each data object appears in at least a single circle. These spheres are subsets of the dataspace. In the second stage, DCBRD executes the DBSCAN algorithm and obtains ε from the overlapped circular regions.

The distribution-based clustering algorithm for mining in large spatial databases (DBCLASD) [78] focuses on addressing the cluster discovery of arbitrary shapes in large spatial datasets and minimizing the requirement of the input parameters. The algorithm introduces these definitions in order to generate its notion of a cluster.

1) NEAREST NEIGHBOR AND NEAREST NEIGHBOR DISTANCE

For an object $x_i \in X$, its nearest neighbor is an object $x_j \in X - \{x_i\}$ which has the minimum distance to the object x_i. The distance from x_i to its nearest neighbor x_j is called the nearest neighbor distance of x_i:

$$Nearest(x_i) = \{x_j \in X : x_j = \min_{x} dist(x_i, x_j), \forall x(x \in X \neq x_i)\} \quad (8)$$

$$NearestDist(x_i) = dist(x_i, Nearest(x_i)). \quad (9)$$

2) NEAREST NEIGHBOR DISTANCE SET

Let X be the set of objects in the dataset and x_n be the elements in X. Then, the nearest neighbor distance set of X, $NearestDistSet(X)$, is defined as the multiset of all values $NearestDist(x_n)$.

The DBCLASD algorithm builds on these definitions and bases its concept of a cluster on analyzing the expected distribution of the distances of objects in a cluster to their nearest neighbor. The objects within a cluster are assumed to be uniformly distributed. The algorithm then expands an initial cluster by adding the nearest neighbor objects as cluster members as long as the newly produced cluster fits an expected distribution. The REGIONQUERY operation is employed to retrieve of neighboring objects. The R*-trees is used as a spatial access method to obtain efficiency.

Locally scaled density-based clustering (LSDBC) [79] is a clustering algorithm that focuses on automating the process of clustering by employing local density information to identify arbitrary clusters in a dataset. In terms of comparing its technique for density estimation, the methods can be classified into two categories. The original DBSCAN algorithm uses the concept of a Parzen window for the purpose of calculating regional density. The Parzen window is a method of computing a hypersphere (hypercube, based on the method used for distance measures) and estimating the regional density by counting the number of objects falling within the computed hypersphere (neighborhood).

The challenge this approach presents is the problem of choosing the most appropriate window size to determine the neighborhood. Alternatively, LSDBC estimates its density through the $k-$nearest neighbors ($kNN-$type) approach. The algorithm starts with the density-based ordering of the objects and applies its cluster expansion method by starting from the densest possible object. The LSDBC determines the cut-off criteria for the expansion process based on the density of

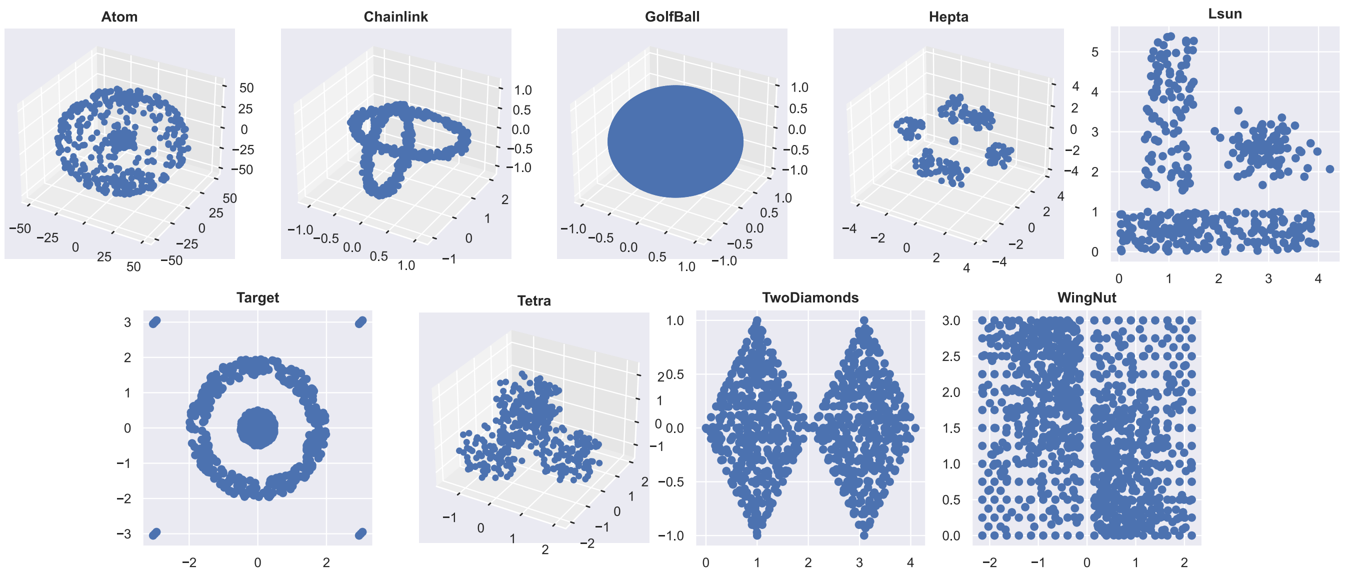

FIGURE 6. **Fundamental Clustering and Projection Suite (FCPS) datasets used for the experiments.**

the center of the cluster, (i.e., expansion proceeds until the density of the generated cluster falls below a prespecified ratio of the center point density). Unlike the DBSCAN algorithm, LSDBC does not expect the input parameters of ε and *minPts*, two arguments used for the Parzen window method to compute the hypersphere volume and determine the regional density. However, the *kNN* method of LSDBC depends on the value of k: the number of the nearest neighbors to consider for each data object during density calculation.

IV. EXPERIMENTS

This section evaluates the experimental results for five different density-based algorithms, each run on two different collection of datasets. Additionally, a density-based subspace clustering algorithm was evaluated using a set of real-world multidimensional datasets. The focus of the experiment is to analyze the quality and performance of the algorithms. For quality measures, the Adjusted Rand Index (ARI) [87], [88] and F-score were used. The ARI is a commonly used for cluster validation because it measures the agreement between two partitions. A larger value of the ARI indicates a higher agreement between two partitions. Moreover, the F-score is an appropriate measurement, as it indicates the accuracy result of a test. The F-score has also been a common measurement scale of a clustering algorithm's accuracy in previous literature in the field. To compare the computation time of the algorithms, we took several partitions of a real-world dataset and examined the total required time for each partitioned dataset to process in order to understand how each algorithm scales up as the instances in a dataset grow.

A. DATASETS

For the experiments to test the accuracy and speed of the various algorithms, we have elected to use one set of synthetic datasets and one real-world dataset. The clustering algorithms were run against the synthetic datasets to test the clustering validation provided by each of the algorithms. The second dataset indicated each algorithm's performance in real-world conditions. The experimental results were used to demonstrate the computational time of each algorithm. The real-world dataset is a 2D dataset, whereas the set of synthetic data contains a combination of 2D and 3D datasets. Furthermore, an additional 10 multidimensional real-world datasets were employed to analyze how the original algorithm fares against SUBCLU, the reimplementation of the DBSCAN algorithm, to cluster datasets with multiple dimensions.

1) FUNDAMENTAL CLUSTERING AND PROJECTION SUITE

The Fundamental Clustering and Projection Suite (FCPS) contains 10 synthetic datasets [89]. These datasets are named ''Atom,'' ''GolfBall,'' ''Hepta,'' ''Chainlink,'' ''EngyTime,'' ''Lsun,'' ''Target,'' ''Tetra,'' ''TwoDiamonds,'' and ''WingNut''. Table 2 describes the instances and dimensions of each of the datasets used for the experiment. Each one was built to challenge clustering algorithms based on different criteria. Each dataset has specific criteria such as the lack of linear separability, class spacing differences, outlier presence, and others. Hence, the table also details these criteria and problems clustering algorithms could face for each dataset in the FCPS. A graphical representation of the FCPS datasets is presented in Fig. 6.

The components of the FCPS each have cluster labels for their data objects. This information was used to compare the data objects' cluster labels to the labels generated by the clustering algorithms during the experiment. The ARI and F-score used these two set of cluster labels to measure the agreement between the labels formed by the clustering algorithms and the labels from the FCPS.

TABLE 2. Description of the dataset in the FCPS.

Name	Instances	Dimensions	Problems
Atom	800	3	Different inner class distances
Chainlink	1000	3	Linearly not separable
GolfBall	4002	3	No cluster structure
Hepta	212	3	Different inner class variances
Lsun	400	2	None
Target	770	2	Presence of outliers
Tetra	400	3	Small inter class distances
TwoDiamonds	800	2	Touching classes
WingNut	1016	2	Density variation within classes

2) GEO-TAGGED TWITTER DATASET

Junjun Yin from the National Center for Supercomputing Application (NCSA) collected the real-world dataset for the experiments [71], [90]. The dataset was obtained through the free Twitter streaming API [91]. The original collection contains exactly 1% of all geo-tagged tweets from the United Kingdom in June 2014 and has about 16.6 million tweets. The subset of the dataset used for the experiments was generated by filtering the dataset to on the first week of June. This filtered dataset contains 3,704,351 instances. Furthermore, in order to analyze how each clustering algorithm performs as the number of instances in the dataset increases, experiments on the dataset were run on several partitions of the dataset, (i.e., at 1,000, 10,000, 50,000, 100,000, 1,000,000 and 2,000,000 instances). The full and filtered dataset can be found on B2SHARE [90].

3) REAL-WORLD MULTIDIMENSIONAL DATASETS

In Sec. III, we discuss how the SUBCLU (density-based subspace clustering) was introduced in order to combat the challenge faced by the DBSCAN algorithm when clustering datasets with high dimensions. To gain an empirical understanding of what SUBCLU had achieved, we collected 10 real-world datasets of various dimensions to compare the clustering results between SUBCLU and the original DBSCAN. Nine of the collected datasets can be found on the UCI Machine Learning Repository [92], and one additional spatial dataset was collected from the data repository at the University of Minnesota [93]. Table 3 describes these datasets in further detail.

B. ALGORITHMS

For the experiments, we collected source codes for the original DBSCAN algorithm [95], revised DBSCAN for adjacent clusters [96], OPTICS [97], HDBSCAN [98], HPDBSCAN [99], and SUBCLU [100]. All the algorithms except SUBCLU are C++ implementations. Whereas DBSCAN, revised DBSCAN, OPTICS, HDBSCAN, and SUBCLU accept a comma-separated value (csv) files as input to process their clustering algorithm, the HPDBCAN algorithm requires its

TABLE 3. Description of the 10 real-world multidimensional datasets.

Name	Instances	Dimensions	Description
Iris [92]	150	4	Types of iris plants
Travel Review [94]	980	10	Reviews on destinations across East Asia
Wine [92]	178	13	Chemical analysis of wines created in Italian regions
MFCC [92]	7195	22	Audio records of different Anuran species calls
Ecoli [92]	336	8	Protein localization sites
Yeast [92]	1484	8	
Glass [92]	214	10	Glass identification data in terms of oxide content
Rock Properties [93]	4121	2	Properties of rocks in Minnesota
WDBC [92]	569	30	Breast cancer Wisconsin (diagnostic) dataset
TAE [92]	151	5	Evaluations of teaching assistants' performance

datasets to be in Hierarchical Data Format v. 5 (HDF5) format [101]. For the neighborhood search for data objects, the DBSCAN, revised DBSCAN, HDBSCAN, and SUBCLU implementations in the experiment perform a linear neighborhood scan, and OPTICS uses an implementation of *kd*-tree [102], whereas HPDBSCAN works with a grid-based data preprocessing and indexing method. This neighborhood search variation helps portray how scalable each implementation is as the dataset increases and provides evidence of the DBSCAN claim of $O(n \log n)$ by implementing such data structures. Due to differences in programming languages, time performance comparison between the SUBCLU and DBSCAN algorithms was not possible. Regardless, the clustering results generated from the corresponding algorithms are worth investigating and were analyzed.

C. RESULTS

The experimental results for each of the experiments are reported here. Table 4 lists the results of each algorithm when run against the FCPS collection. Each table provides the necessary input parameters provided for the algorithms. An algorithm's corresponding input parameter for a particular dataset is the parameter that returns the best results for the clustering algorithm. Where the DBSCAN, revised DBSCAN and HPDBSCAN take two parameters (*minPts* and ε), the OPTICS and HDBSCAN take only *minPts* as input. All experiments were conducted on two Intel Xeon CPU E5-2695 processors clocked at 2.10 GHz.

To evaluate the cluster accuracy produced by the algorithms, we have measured the ARI and F-score. The ARI is a variation on the classic Rand Index method. The ARI expresses the ratio of the cluster assignments regarded as "correct." It calculates a similarity measure between two different clusters by considering all pairs of samples, then

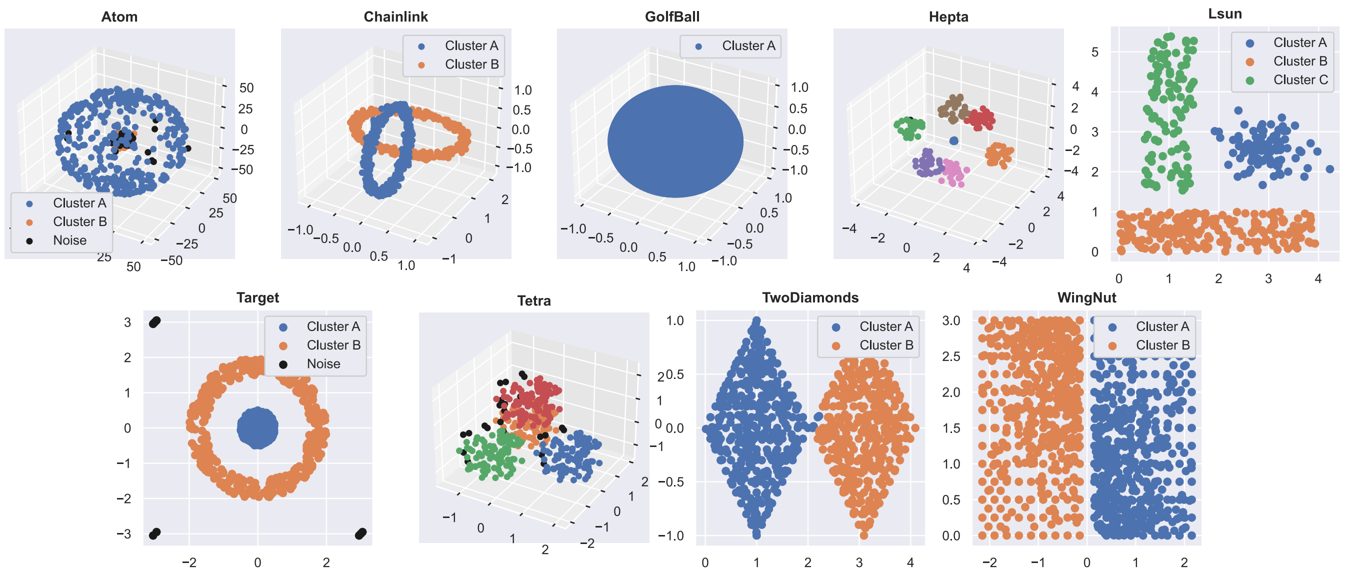

FIGURE 7. **Clusters generated from DBSCAN on the fundamental clustering and projection suite datasets.**

counting the pairs assigned in the same or in the different clusters produced, to finally compare against the "true" clusters, adjusting for random chance. The ARI is bounded between -1 and 1. The F-score is calculated from the precision and recall of a test, where precision is the ratio of true positive results to the total positive results found, and recall is the ratio of true positive results to the actual positive labels. The final F-score value is computed by calculating the harmonic mean of the precision and recall. For both the ARI and F-score measures, the similarity is better as the value approaches 1. We evaluated the cluster results using the scikit-learn [103] implementation of the ARI and F-score.

1) FCPS EXPERIMENTAL RESULTS

From the collection of 3D and 2D datasets of the FCPS, the DBSCAN generally performs well to find the appropriate clusters of each dataset. The "Lsun," "Chainlink," "Target," "WingNut," and "GolfBall" datasets were clustered with perfect accuracy by the DBSCAN algorithm. Furthermore, the DBSCAN algorithm correctly estimated the noise objects present in the "Target" dataset.

This accuracy can be observed in the table because the DBSCAN algorithm generated an accuracy of 100% when clustering "Target," although only 98.5% of the data objects had been assigned a cluster. For the other datasets (i.e., "Hepta," "Tetra," "Atom," and "TwoDiamonds"), DBSCAN did not correctly cluster all data objects. "TwoDiamonds" is evidence of a particular case in which the DBSCAN managed to group all the data objects into a cluster, although the assignment was not always to the correct cluster. Fig. 7 presents the clustering outcome of the DBSCAN algorithm for the FCPS and how it is challenged by datasets, such as "TwoDiamonds" where two or more different clusters appear close to each other.

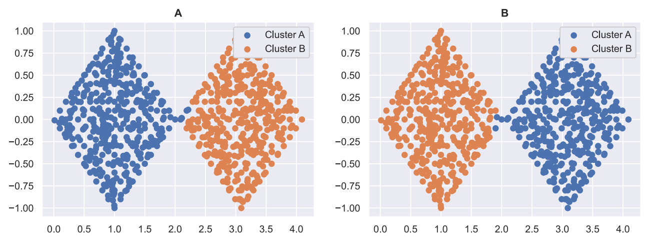

FIGURE 8. **Two different runs of the DBSCAN on the same dataset with different results. The DBSCAN expands clusters through the density-reachability mechanism, so cluster labels for border objects of adjacent clusters might vary depending on which data objects are being randomly processed first. (A) Initialized the cluster expansion from a certain data object located in the left "Diamond." (B) Started from a data object found in the right "Diamond".**

The results gathered for the revised DBSCAN algorithm provide an interesting comparison with the original DBSCAN results. The most significant differences are observed in the "Tetra" and "TwoDiamonds" datasets; datasets where the clusters are formed adjacent to one another. The adjacency of clusters appears when border objects that correspond to one cluster also appear in the neighborhood region of another cluster. The original DBSCAN's assignment of cluster labels for such border objects is determined by its random iteration of the dataset (i.e., if a particular border object belongs to the region of two different clusters, DBSCAN automatically assigns that border object to the cluster it constructs first, even before considering the second cluster). Furthermore, because DBSCAN chooses data objects randomly in order to start building clusters, multiple runs of the algorithm on the same data object could produce different clusters due to the assignments of the border objects as shown in Fig. 8. This condition is displayed in the "TwoDiamonds" and "Tetra"

TABLE 4. Experimental results on the fundamental clustering and projection suite.

Algorithms	Datasets			Parameters [*]		Clusters Found	% of Objects Clustered	Time Elapsed (sec)	F-Score	Adjusted Rand Index
	Name	Instances	Dimensions	$minPts$	ε					
DBSCAN (Density-based spatial clustering applications with noise)	Hepta	212	3	4	0.75	7	99.53	0.008217	0.995	0.9942
	Lsun	400	2	4	0.5	3	100	0.017831	1	1
	Tetra	400	3	4	0.4	4	94	0.019431	0.832	0.665
	Chainlink	1000	3	4	0.15	2	100	0.080062	1	1
	Atom	800	3	7	15	2	98.875	0.065361	0.988	0.978
	Target	770	2	4	0.25	2	98.44	0.044343	1	1
	TwoDiamonds	800	2	50	0.4	2	100	0.044873	0.941	0.779
	WingNut	1016	2	4	0.25	2	100	0.069513	1	1
	GolfBall	4002	3	4	0.07	1	100	0.742973	1	1
Revised DBSCAN algorithm to cluster data with dense adjacent clusters	Hepta	212	3	4	0.75	7	99.53	0.009209	0.995	0.9942
	Lsun	400	2	4	0.5	3	100	0.025487	1	1
	Tetra	400	3	4	0.4	4	94	0.021018	0.939	0.914
	Chainlink	1000	3	4	0.15	2	100	0.113414	1	1
	Atom	800	3	7	15	2	98.875	0.092229	0.988	0.978
	Target	770	2	4	0.25	2	98.44	0.0736	1	1
	TwoDiamonds	800	2	50	0.4	2	98.25	0.077226	1	1
	WingNut	1016	2	4	0.25	2	100	0.110064	1	1
	GolfBall	4002	3	4	0.07	1	100	0.727785	1	1
HPDBSCAN (Highly parallel DBSCAN)	Hepta	212	3	4	0.75	7	99.53	0.001675	0.995	0.9942
	Lsun	400	2	4	0.5	3	100	0.001435	1	1
	Tetra	400	3	4	0.4	4	94	0.002675	0.832	0.665
	Chainlink	1000	3	4	0.15	2	100	0.003679	1	1
	Atom	800	3	7	15	2	98.875	0.005633	0.988	0.978
	Target	770	2	4	0.25	2	98.44	0.003321	1	1
	TwoDiamonds	800	2	50	0.4	2	100	0.004075	0.941	0.779
	WingNut	1016	2	4	0.25	2	100	0.003614	1	1
	GolfBall	4002	3	4	0.07	1	100	0.025764	1	1
OPTICS (Ordering points to identify the clustering structures)	Hepta	212	3	4		7	100	0.051152	1	1
	Lsun	400	2	4		3	100	0.06457	1	1
	Tetra	400	3	4		4	94.75	0.077324	0.925	0.8766
	Chainlink	1000	3	4		2	100	0.27029	1	1
	Atom	800	3	7		2	98.875	0.217918	0.988	0.978
	Target	770	2	6		2	98.44	0.101592	1	1
	TwoDiamonds	800	2	6		2	100	0.124657	0.941	0.779
	WingNut	1016	2	4		2	100	0.102829	1	1
	GolfBall	4002	3	4		1	100	0.395772	1	1
HDBSCAN (Density-based clustering based on hierarchical density estimates)	Hepta	212	3	4		7	100	2.64	1	1
	Lsun	400	2	4		3	100	4.07	1	1
	Tetra	400	3	4		4	91.5	4.01	0.925	0.8766
	Chainlink	1000	3	4		2	100	7.67	1	1
	Atom	800	3	7		2	100	8.48	1	1
	Target	770	2	4		3	100	7.37	1	1
	TwoDiamonds	800	2	6		3	98	7.08	0.99	0.9603
	WingNut	1016	2	4		4	99.8	8.13	0.99	0.996
	GolfBall	4002	3	4		3	100	40.56	-	-

[*] OPTICS and HDBSCAN only take $minPts$ as Parameter, ε is not used.

datasets of the FCPS, where data objects belonging to one cluster were clustered into another.

The revised DBSCAN algorithm provides an improved implementation for such cases. Instead of assigning a border object to a particular cluster as soon as it is detected, the revised DBSCAN algorithm first constructs clusters by only considering their core-object members. Then, this algorithm determines which core object is closest for each border object and assigns it the cluster label of that core object. This implementation ensures the correct assignment of cluster labels for all border objects, as illustrated on Fig. 9.

Additionally, consistent cluster shapes and cluster labels at every run of the algorithm were observed. The gains in accuracy when clustering "TwoDiamonds" and "Tetra" with the revised DBSCAN corroborate this implementation. "TwoDiamonds" is perfectly clustered with the revised DBSCAN, while the accuracy of "Tetra" increases by 37% to 0.914 as presented on Table 4. The time penalty for the additional computation of the revised DBSCAN algorithm is also evident, but a more consequential observation of this can be made on the Twitter dataset.

The HPDBSCAN implementation divides the iterative computation required for the EXPANDCLUSTER phase of

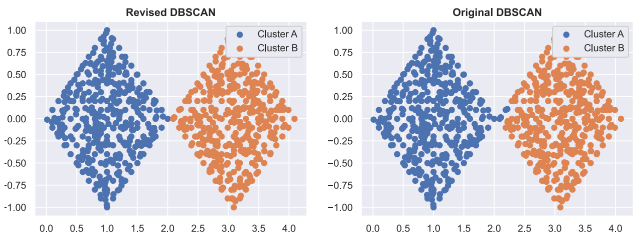

FIGURE 9. TwoDiamonds cluster results for the revised DBSCAN vs. the original DBSCAN algorithm. The revised DBSCAN (left) correctly labels the border objects of every cluster and ensures all runs of the algorithm return the same shape of the clusters.

TABLE 5. Computational time for the five algorithms on the Twitter dataset (in seconds).

Execution Time in seconds						
Algorithms	Twitter Dataset Instances					
	1000	*10,000*	*50,000*	*100,000*	*1,000,000*	*2,000,000*
DBSCAN	0.063914	3.48978	87.467	353.202	34847.4	138276
Revised DBSCAN	0.077539	4.3934	109.711	462.188	51018.3	199842
HDBSCAN	0.459	101.216	3169.252	24646.342	-	-
OPTICS	0.208421	4.99076	63.9069	207.396	14941.3	56953
HPDBSCAN	0.06204	0.074709	0.127163	0.19887	1.505743	3.815985

the algorithm among the 72 threads of the two compute nodes available for the experiment. This implementation results in a significant acceleration in computational time as compared to the original DBSCAN algorithm, as listed in Table 4. Although the change in this scalability is better observed on larger datasets, such as the Twitter data, the FCPS collection also exhibits time differences between the HPDBSCAN and the original DBSCAN. The ''GolfBall'' dataset portrays this most significantly, from 0.743 s for DBSCAN to 0.026 s for HPDBSCAN, a time decrease of 96.53%.

2) TWITTER DATASET EXPERIMENTAL RESULTS

For the Twitter dataset experiment, subsets of the Twitter dataset at various instances were analyzed to determine how each algorithm scales as the size of the dataset increases. This experiment focuses on the computational time of the algorithms, as no predefined cluster labels exist for each data object to compare with using the ARI method. The input parameters of *minPts* and ε were chosen at 40 and 0.01 respectively [71]. Table 5 presents the time taken for each algorithm to cluster the dataset at subsets of 1,000, 10,000, 50,000, 100,000, 1,000,000 and 2,000,000 instances of the Twitter dataset.

The DBSCAN algorithm used for this experiment executes a brute force implementation of the neighborhood search for its REGIONQUERY phase, which corresponds to a time complexity of $O(n)$ per regional query. As such, the experiment conducted on the DBSCAN exhibits a quadratic increase with respect to the number of instances.

The revised DBSCAN performs slightly slower than the DBSCAN due to its added procedure of assigning border objects to their respective clusters. However, the OPTICS

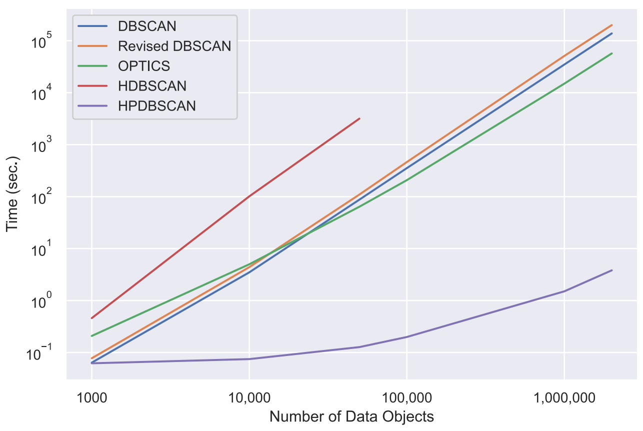

FIGURE 10. Scalability of the computational time of five algorithms as instances of the Twitter dataset increases exponentially.

algorithm with the *kd*-tree implementation performs its REGIONQUERY at $O(\log n)$ and the results corroborate this with a faster time than the DBSCAN's implementation. The HDBSCAN algorithm required more memory space than that available for the instances of $\geq$1,000,000 at the given input parameter of $minPts = 40$. For the sake of fairness of the experiment, we chose not to alter to a more manageable *minPts* for the HDBSCAN and reported only the results of instances of <1,000,000. Finally, the HPDBSCAN's parallel implementation of the originally sequential DBSCAN algorithm results in a significant drop in time, with the data instances of 2,000,000 requiring only 3.82 s to finish its run. Fig. 10 plots the results of the experiment in order to visualize scalability of the five algorithms. At the time of writing, to our best knowledge, HPDBSCAN is the only C++ implementation of parallel DBSCAN publicly available for our evaluation; however, it is worth investigating how other parallel implementations of DBSCAN compare to it.

3) MULTIDIMENSIONAL DATASETS EXPERIMENTAL RESULTS

Further, 10 real-world datasets exhibiting varied dimensionality were examined by running on the original DBSCAN and the SUBCLU clustering algorithms. The SUBCLU clustering algorithm, as discussed in Sec. III works by iterating through the dimension subsets (subspaces) of the dataset in order to locate clusters hidden deep within the feature space. It also incorporates a pruning mechanism to reduce the search space by applying the monotonicity property of density-connected sets in a bottom-up approach to subspace clustering. As such, multiple, usually overlapping clusters in different subspaces are generated when running the SUBCLU algorithm as opposed to the cluster output of the original DBSCAN in a single search space. For the experimental results, the number of clusters and coverage percentage reported for the SUBCLU algorithm are those from subspaces that return the greatest number of clusters and objects with the least amount of noise.

TABLE 6. Experimental results on 10 multidimensional real-world datasets.

Algorithm	Dataset	Shape		Parameters		Clusters Found [*]	% Clustered
		Instances	Dimensions	ε	$minPts$		
DBSCAN (Density-Based Spatial Clustering Application with Noise)	Iris	150	4	0.9	8	2	100
	Travel Review	980	10	0.67	20	2	71.8
	Wine	178	13	51	26	2	48.9
	MFCC	7195	22	1.4	44	3	99.7
	Ecoli	336	8	0.2	16	2	86.01
	Yeast	1484	8	0.09	16	2	40.9
	Glass	214	10	2	20	2	87.4
	Rock Properties	4121	2	25000	4	10	99.8
	WDBC	569	30	No Cluster Structure			
	TAE	151	5	0.5	10	3	100
SUBCLU (Density-Based Subspace Clustering)	Iris	150	4	0.9	8	2	100
	Travel Review	980	10	0.67	20	2	86.3
	Wine	178	13	51	26	2	48.9
	MFCC	7195	22	1.4	44	4	92.9
	Ecoli	336	8	0.2	16	2	96.13
	Yeast	1484	8	0.09	16	2	40.9
	Glass	214	10	2	20	2	87.4
	Rock Properties	4121	2	25000	4	10	99.8
	WDBC	569	30	No Cluster Structure			
	TAE	151	5	0.5	10	3	100

[*] Cluster results for SUBCLU reported from the subspaces with the most clusters consisting of greater coverage found.

Furthermore, to identify the appropriate parameter values for the experiments, the approaches proposed in [41] and [56] were implemented. These papers suggest a *minPts* that corresponds to twice the number of dimensions present in the dataset; and use the calculated *minPts* to plot the *kNN* plot of the dataset. The ε corresponds to the plot where the maximum curvature is revealed. Table 6 lists the clustering results generated for the two algorithms. The *minPts* values used in each of the reported dataset is double the value of its dimension.

Of the ten datasets used to conduct the experiment, three display varying cluster results. The "Travel Review" and "Ecoli" datasets indicate greater coverage of clustered objects, and "MFCC" was able to identify additional cluster in the SUBCLU algorithm than the original DBSCAN algorithm. Meanwhile, the "WDBC" dataset, which returned a single cluster when appropriate parameters were used, demonstrated no improvement when run against SUBCLU. This result suggests the subspace clustering method was not sufficiently adept at identifying significant clusters within the subsets of the feature spaces of the datasets. The most likely reason for this scenario is the rigidity of the parameters used during the SUBCLU procedure. Although the algorithm examines the subspaces of the datasets, a preset value of ε and *minPts* was used on each of the subspaces, which limits the algorithm's ability to identify whether a particular feature space holds meaningful clusters or not, if it was run using a more subspace-appropriate parameter. Therefore, the SUBCLU algorithm could either prune the subspace and all its successive subspaces generated in a bottom-up approach or generate more noise objects where there should not be. Future DBSCAN-based clustering algorithms can produce a more fluid parameter generation method for subspace clustering in order to address this problem and reduce the over-reliance of preset values for the SUBCLU algorithm.

V. CONCLUSION

Several density-based algorithms have been proposed that aim to improve on the original DBSCAN method. Based on their implementation, these algorithms can be categorized as focusing on either improving clustering quality, optimizing efficiency, or combating the reliance on parameter settings. The OPTICS and HDBSCAN algorithms produce a hierarchy of clusters that extract a densely collected set of objects within a cluster. While OPTICS can identify the denser clusters using a reachability plot, it does not provide an alternative means to identify a flat partitioning of the most significant clusters without using the original DBSCAN approach. The revised DBSCAN also aims to improve on clustering quality by focusing on datasets that exhibit adjacent clusters. It successfully identified the correct cluster labels for border objects appearing in two different cluster neighborhoods. However, this added implementation increases the computational time from the experiments, and datasets that do not have the adjacency property would suffer without gains in cluster quality. A "highly parallel" DBSCAN (HPDBSCAN) that employs OpenMPI/MPI to break the sequential process of the original algorithm and accelerate neighborhood searches in a distributed processing environment was also evaluated. This implementation was tested on a real dataset and was capable of executing within a time complexity that is orders of magnitude faster than the other algorithms.

Experiments on three collection of datasets were conducted. The results from the FCPS collection reveal that, while the improved algorithms provide better quality results in particular cases, the original DBSCAN is generally adept at handling clustering operations. The real-world Twitter dataset used to test the efficiency of the algorithms acknowledged HPDBSCAN's parallel implementation of the clustering while providing support for adapting indexing structures such as the R*-tree or *kd*-tree for faster regional queries. Finally, the 10 real-world datasets collected to test how density-based clustering algorithms handle multidimensional feature spaces indicated the high sensitivity of SUBCLU's results to the input parameters ε and *minPts*.

REFERENCES

[1] X. Wang and J. M. Garibaldi, "A comparison of fuzzy and non-fuzzy clustering techniques in cancer diagnosis," in *Proc. 2nd Int. Conf. Comput. Intell. Med. Healthcare, BIOPATTERN Conf.*, Lisbon, Portugal, vol. 28, 2005.

[2] J. Liu and R. Han, "Spectral clustering and multicriteria decision for design of district metered areas," *J. Water Resour. Planning Manage.*, vol. 144, no. 5, May 2018, Art. no. 04018013.

[3] O. J. Oyelade, O. O. Oladipupo, and I. C. Obagbuwa, "Application of k means clustering algorithm for prediction of students academic performance," 2010, *arXiv:1002.2425*. [Online]. Available: http://arxiv.org/abs/1002.2425

[4] K. Y. Yeung, M. Medvedovic, and R. E. Bumgarner, "Clustering gene-expression data with repeated measurements," *Genome Biol.*, vol. 4, no. 5, p. R34, 2003.

[5] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein, "Cluster analysis and display of genome-wide expression patterns," *Proc. Nat. Acad. Sci. USA*, vol. 95, no. 25, pp. 14863–14868, Dec. 1998.

[6] K. Akkaya, F. Senel, and B. McLaughlan, "Clustering of wireless sensor and actor networks based on sensor distribution and connectivity," *J. Parallel Distrib. Comput.*, vol. 69, no. 6, pp. 573–587, Jun. 2009.

[7] A. Saurabh and A. Naik, "Wireless sensor network based adaptive landmine detection algorithm," in *Proc. 3rd Int. Conf. Electron. Comput. Technol.*, vol. 1, Apr. 2011, pp. 220–224.

[8] S. Alelyani, J. Tang, and H. Liu, "Feature selection for clustering: A review," *Data clustering: Algorithms Appl.*, vol. 29, no. 1, pp. 29–60, 2013.

[9] L. Song, A. Smola, A. Gretton, K. M. Borgwardt, and J. Bedo, "Supervised feature selection via dependence estimation," in *Proc. 24th Int. Conf. Mach. Learn. (ICML)*, 2007, pp. 823–830.

[10] J. Weston, A. Elisseeff, B. Schölkopf, and M. Tipping, "Use of the zero-norm with linear models and kernel methods," *J. Mach. Learn. Res.*, vol. 3, pp. 1439–1461, Mar. 2003.

[11] J. G. Dy and C. E. Brodley, "Feature selection for unsupervised learning," *J. Mach. Learn. Res.*, vol. 5, pp. 845–889, Jan. 2004.

[12] P. Mitra, C. A. Murthy, and S. K. Pal, "Unsupervised feature selection using feature similarity," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 3, pp. 301–312, Mar. 2002.

[13] A. Saxena, M. Prasad, A. Gupta, N. Bharill, O. P. Patel, A. Tiwari, M. J. Er, W. Ding, and C.-T. Lin, "A review of clustering techniques and developments," *Neurocomputing*, vol. 267, pp. 664–681, Dec. 2017.

[14] L. Rokach and O. Maimon, "Clustering methods," in *Data Mining and Knowledge Discovery Handbook*. Boston, MA, USA: Springer, 2005, pp. 321–352.

[15] A. R. Webb, *Statistical Pattern Recognition*. Hoboken, NJ, USA: Wiley, 2003.

[16] A. K. Jain, R. P. W. Duin, and J. Mao, "Statistical pattern recognition: A review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 1, pp. 4–37, Jan. 2000.

[17] G. Gan, C. Ma, and J. Wu, *Data Clustering: Theory, Algorithms, and Applications*. Philadelphia, PA, USA: SIAM, 2020.

[18] C. C. Aggarwal and C. Reddy, *An Introduction to Cluster Analysis*, 1st ed. Boca Raton, FL, USA: Chapman & Hall, 2013.

[19] P.-E. Danielsson, "Euclidean distance mapping," *Comput. Graph. Image Process.*, vol. 14, no. 3, pp. 227–248, Nov. 1980.

[20] D. T. Nguyen, L. Chen, and C. K. Chan, "Clustering with multiviewpoint-based similarity measure," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 6, pp. 988–1001, Jun. 2012.

[21] K. Pearson, "VII. Mathematical contributions to the theory of evolution.—III. Regression, heredity, and panmixia," *Phil. Trans. Roy. Soc. London A, Containing Papers Math. Phys. Character*, vol. 187, pp. 253–318, Dec. 1896.

[22] A. Bravais, *Analyse Mathématique sur les Probabilités des Erreurs de Situation d'un Point*. Paris, France: Impr. Royale, 1844.

[23] P. Arabie and L. Hubert, "Advances in cluster analysis relevant to marketing research," in *From Data to Knowledge*. Berlin, Germany: Springer, 1996, pp. 3–19.

[24] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognit. Lett.*, vol. 31, no. 8, pp. 651–666, Jun. 2010.

[25] A. Sharma and V. Rastogi, "Spam filtering using k mean clustering with local feature selection classifier," *Int. J. Comput. Appl.*, vol. 108, no. 10, pp. 35–39, Dec. 2014.

[26] S. Hosseinimotlagh and E. E. Papalexakis, "Unsupervised content-based identification of fake news articles with tensor decomposition ensembles," in *Proc. Workshop Misinformation Misbehavior Mining Web (MIS2)*, 2018.

[27] V. Estivill-Castro and J. Yang, "Fast and robust general purpose clustering algorithms," in *Proc. Pacific Rim Int. Conf. Artif. Intell.* Berlin, Germany: Springer, 2000, pp. 208–218.

[28] C. Fraley, "How many clusters? Which clustering method? Answers via model-based cluster analysis," *Comput. J.*, vol. 41, no. 8, pp. 578–588, Aug. 1998.

[29] J. Han, J. Pei, and M. Kamber, *Data Mining: Concepts and Techniques*. Amsterdam, The Netherlands: Elsevier, 2011.

[30] A. Baraldi and P. Blonda, "A survey of fuzzy clustering algorithms for pattern recognition. I," *IEEE Trans. Syst., Man Cybern. B, Cybern.*, vol. 29, no. 6, pp. 778–785, Dec. 1999.

[31] A. Baraldi and P. Blonda, "A survey of fuzzy clustering algorithms for pattern recognition. II," *IEEE Trans. Syst., Man Cybern. B, Cybern.*, vol. 29, no. 6, pp. 786–801, Dec. 1999.

[32] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A k-means clustering algorithm," *Appl. Statist.*, vol. 28, no. 1, pp. 100–108, 1979.

[33] C. K. Reddy and B. Vinzamuri, "A survey of partitional and hierarchical clustering algorithms," in *Data Clustering: Algorithms and Applications*. Boca Raton, FL, USA: Chapman & Hall, Sep. 2013, pp. 87–110.

[34] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: Hierarchical clustering using dynamic modeling," *Computer*, vol. 32, no. 8, pp. 68–75, 1999.

[35] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," *ACM SIGMOD Rec.*, vol. 25, no. 2, pp. 103–114, Jun. 1996.

[36] W. Cheng, W. Wang, and S. Batista, "Grid-based clustering," in *Data Clustering*. Boca Raton, FL, USA: CRC Press, 2018, pp. 128–148.

[37] W. Wang, J. Yang, and R. Muntz, "STING: A statistical information grid approach to spatial data mining," in *Proc. VLDB*, vol. 97, 1997, pp. 186–195.

[38] G. Sheikholeslami, S. Chatterjee, and A. Zhang, "Wavecluster: A multi-resolution clustering approach for very large spatial databases," in *Proc. 24th Int. Conf. Very Large Databases*, vol. 98, Aug. 1998, pp. 428–439.

[39] B. Andreopoulos, "Clustering categorical data," in *Wiley StatsRef: Statistics Reference Online*. Boca Raton, FL, USA: Chapman & Hall, 2014, pp. 1–12.

[40] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. KDD*, 1996, vol. 96, no. 34, pp. 226–231.

[41] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "DBSCAN revisited, revisited: Why and how you should (Still) use DBSCAN," *ACM Trans. Database Syst.*, vol. 42, no. 3, pp. 1–21, Aug. 2017.

[42] H.-P. Kriegel, P. Kröger, J. Sander, and A. Zimek, "Density-based clustering," *Wiley Interdiscipl. Rev., Data Mining Knowl. Discovery*, vol. 1, no. 3, pp. 231–240, May/Jun. 2011.

[43] D. P. de Oliveira, J. H. Garrett, and L. Soibelman, "A density-based spatial clustering approach for defining local indicators of drinking water distribution pipe breakage," *Adv. Eng. Informat.*, vol. 25, no. 2, pp. 380–389, Apr. 2011.

[44] M. Daszykowski, B. Walczak, and D. L. Massart, "Looking for natural patterns in data: Part 1. density-based approach," *Chemometrics Intell. Lab. Syst.*, vol. 56, no. 2, pp. 83–92, 2001.

[45] L. Zhou, P. K. Hopke, and P. Venkatachari, "Cluster analysis of single particle mass spectra measured at flushing, NY," *Analytica Chim. Acta*, vol. 555, no. 1, pp. 47–56, Jan. 2006.

[46] S. Liu, Z.-T. Dou, F. Li, and Y.-L. Huang, "A new ant colony clustering algorithm based on DBSCAN," in *Proc. Int. Conf. Mach. Learn. Cybern.*, vol. 3, 2004, pp. 1491–1496.

[47] A. Ghosh, A. Halder, M. Kothari, and S. Ghosh, "Aggregation pheromone density based data clustering," *Inf. Sci.*, vol. 178, no. 13, pp. 2816–2831, Jul. 2008.

[48] C. Plant, S. J. Teipel, A. Oswald, C. Böhm, T. Meindl, J. Mourao-Miranda, A. W. Bokde, H. Hampel, and M. Ewers, "Automated detection of brain atrophy patterns based on MRI for the prediction of Alzheimer's disease," *NeuroImage*, vol. 50, no. 1, pp. 162–174, Mar. 2010.

[49] J. Gong and C. H. Caldas, "Data processing for real-time construction site spatial modeling," *Autom. Construct.*, vol. 17, no. 5, pp. 526–535, Jul. 2008.

[50] T. N. Tran, K. Drab, and M. Daszykowski, "Revised DBSCAN algorithm to cluster data with dense adjacent clusters," *Chemometric Intell. Lab. Syst.*, vol. 120, pp. 92–96, Jan. 2013.

[51] H. Chebi, D. Acheli, and M. Kesraoui, "Dynamic detection of abnormalities in video analysis of crowd behavior with DBSCAN and neural networks," *Adv. Sci., Technol. Eng. Syst. J.*, vol. 1, no. 5, pp. 56–63, Oct. 2016.

[52] H. Li, J. Liu, K. Wu, Z. Yang, R. W. Liu, and N. Xiong, "Spatio-temporal vessel trajectory clustering based on data mapping and density," *IEEE Access*, vol. 6, pp. 58939–58954, 2018.

[53] H. Li, J. Liu, Z. Yang, R. W. Liu, K. Wu, and Y. Wan, "Adaptively constrained dynamic time warping for time series classification and clustering," *Inf. Sci.*, vol. 534, pp. 97–116, Sep. 2020.

[54] R. W. Liu, J. Nie, S. Garg, Z. Xiong, Y. Zhang, and M. S. Hossain, "Data-driven trajectory quality improvement for promoting intelligent vessel traffic services in 6G-enabled maritime IoT systems," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5374–5385, Apr. 2021.

[55] K. Sheridan, T. G. Puranik, E. Mangortey, O. J. Pinon-Fischer, M. Kirby, and D. N. Mavris, "An application of dbscan clustering for flight anomaly detection during the approach phase," in *Proc. AIAA Scitech Forum*, 2020, p. 1851.

[56] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu, "Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications," *Data Mining Knowl. Discovery*, vol. 2, no. 2, pp. 169–194, Jun. 1998.

[57] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: Ordering points to identify the clustering structure," *ACM SIGMOD. Rec.*, vol. 28, no. 2, pp. 49–60, Jun. 1999.

[58] K. Kailing, H.-P. Kriegel, and P. Kröger, "Density-connected subspace clustering for high-dimensional data," in *Proc. SIAM Int. Conf. Data Mining*. SIAM, Apr. 2004, pp. 246–256.

[59] D. Birant and A. Kut, "ST-DBSCAN: An algorithm for clustering spatial–temporal data," *Data Knowl. Eng.*, vol. 60, no. 1, pp. 208–221, Jan. 2007.

[60] P. Liu, D. Zhou, and N. Wu, "VDBSCAN: Varied density based spatial clustering of applications with noise," in *Proc. Int. Conf. Service Syst. Service Manage.*, Jun. 2007, pp. 1–4.

[61] A. Ram, A. Sharma, A. S. Jalal, A. Agrawal, and R. Singh, "An enhanced density based spatial clustering of applications with noise," in *Proc. IEEE Int. Advance Comput. Conf.*, Mar. 2009, pp. 1475–1478.

[62] W. Ashour and S. Sunoallah, "Multi density DBSCAN," in *Proc. Int. Conf. Intell. Data Eng. Automated Learn.* Berlin, Germany: Springer, 2011, pp. 446–453.

[63] H. Darong and W. Peng, "Grid-based DBSCAN algorithm with referential parameters," *Phys. Procedia*, vol. 24, pp. 1166–1170, Jan. 2012.

[64] R. J. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in *Proc. Pacific–Asia Conf. Knowl. Discovery Data Mining*. Berlin, Germany: Springer, 2013, pp. 160–172.

[65] A. Hinneburg and D. A. Keim, "An efficient approach to clustering in large multimedia databases with noise," KDD, Tech. Rep., 1998, pp. 58–65, vol. 98.

[66] B. Liu, "A fast density-based clustering algorithm for large databases," in *Proc. Int. Conf. Mach. Learn. Cybern.*, 2006, pp. 996–1000.

[67] C. Xiaoyun, M. Yufang, Z. Yan, and W. Ping, "GMDBSCAN: Multi-density DBSCAN cluster based on grid," in *Proc. IEEE Int. Conf. e-Business Eng.*, Oct. 2008, pp. 780–783.

[68] P. Viswanath and V. S. Babu, "Rough-DBSCAN: A fast hybrid density based clustering method for large data sets," *Pattern Recognit. Lett.*, vol. 30, no. 16, pp. 1477–1488, Dec. 2009.

[69] C. Böhm, R. Noll, C. Plant, and B. Wackersreuther, "Density-based clustering using graphics processors," in *Proc. 18th ACM Conf. Inf. Knowl. Manage. (CIKM)*, 2009, pp. 661–670.

[70] Y. He, H. Tan, W. Luo, S. Feng, and J. Fan, "MR-DBSCAN: A scalable MapReduce-based DBSCAN algorithm for heavily skewed data," *Frontiers Comput. Sci.*, vol. 8, no. 1, pp. 83–99, Feb. 2014.

[71] M. Götz, C. Bodenstein, and M. Riedel, "HPDBSCAN: Highly parallel DBSCAN," in *Proc. Workshop Mach. Learn. High-Perform. Comput. Environ.*, 2015, pp. 1–10.

[72] L. Dagum and R. Menon, "OpenMP: An industry standard API for shared-memory programming," *IEEE Comput. Sci. Eng.*, vol. 5, no. 1, pp. 46–55, Jan. 1998.

[73] K. M. Kumar and A. R. M. Reddy, "A fast DBSCAN clustering algorithm by accelerating neighbor searching using groups method," *Pattern Recognit.*, vol. 58, pp. 39–48, Oct. 2016.

[74] J. Jang and H. Jiang, "DBSCAN++: Towards fast and scalable density clustering," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 3019–3029.

[75] Y. Chen, L. Zhou, N. Bouguila, C. Wang, Y. Chen, and J. Du, "BLOCK-DBSCAN: Fast clustering for large scale data," *Pattern Recognit.*, vol. 109, Jan. 2021, Art. no. 107624.

[76] S.-S. Li, "An improved DBSCAN algorithm based on the neighbor similarity and fast nearest neighbor query," *IEEE Access*, vol. 8, pp. 47468–47476, 2020.

[77] Y. Wang, Y. Gu, and J. Shun, "Theoretically-efficient and practical parallel DBSCAN," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Jun. 2020, pp. 2555–2571.

[78] X. Xu, M. Ester, H.-P. Kriegel, and J. Sander, "A distribution-based clustering algorithm for mining in large spatial databases," in *Proc. 14th Int. Conf. Data Eng.*, 1998, pp. 324–331.

[79] E. Biçici and D. Yuret, "Locally scaled density based clustering," in *Proc. Int. Conf. Adapt. Natural Comput. Algorithms*. Berlin, Germany: Springer, 2007, pp. 739–748.

[80] A. Fahim, A. Salem, F. Torkey, and M. Ramadan, "Density clustering based on radius of data (DCBRD)," *Int. J. Comput. Inf. Eng.*, 2006.

[81] S. Vijayalaksmi and M. Punithavalli, "A fast approach to clustering datasets using DBSCAN and pruning algorithms," *Int. J. Comput. Appl.*, vol. 60, no. 14, pp. 1–7, Dec. 2012.

[82] M. M. R. Khan, M. A. B. Siddique, R. B. Arif, and M. R. Oishe, "ADBSCAN: Adaptive density-based spatial clustering of applications with noise for identifying clusters with varying densities," in *Proc. 4th Int. Conf. Electr. Eng. Inf. Commun. Technol. (iCEEiCT)*, Sep. 2018, pp. 107–111.

[83] A. Starczewski, P. Goetzen, and M. J. Er, "A new method for automatic determining of the DBSCAN parameters," *J. Artif. Intell. Soft Comput. Res.*, vol. 10, no. 3, pp. 209–221, Jul. 2020.

[84] I. T. Jolliffe, "Principal component analysis," *Technometrics*, vol. 45, no. 3, p. 276, 2003.

[85] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic subspace clustering of high dimensional data for data mining applications," *ACM SIGMOD Rec.*, vol. 27, no. 2, pp. 94–105, Jun. 1998.

[86] H. Späth, "Cluster analysis algorithms for data reduction and classification of objects," Ellis Horwood, Chichester, U.K., Tech. Rep., 1980.

[87] D. Steinley, "Properties of the Hubert-Arable adjusted rand index," *Psychol. Methods*, vol. 9, no. 3, p. 386, 2004.

[88] K. Y. Yeung and W. L. Ruzzo, "Details of the adjusted rand index and clustering algorithms, supplement to the paper an empirical study on principal component analysis for clustering gene expression data," *Bioinformatics*, vol. 17, no. 9, pp. 763–774, May 2001.

[89] M. C. Thrun and A. Ultsch, "Clustering benchmark datasets exploiting the fundamental clustering problems," *Data Brief*, vol. 30, Jun. 2020, Art. no. 105501.

[90] *B2share, HPDBSCAN Benchmark Test Files*. (Accessed:Jan. 16, 2021). [Online]. Available: https://b2share.eudat.eu/records/7f0c22ba9a5a44ca83cdf4fb304ce44e

[91] *Stream Tweets in Real-Time | Docs | Twitter Developer*. (Accessed: Jan. 16, 2021). [Online]. Available: https://developer.twitter.com/en/docs/tutorials/stream-tweets-in-real-time

[92] D. Dua and C. Graff. (2017). *UCI Machine Learning Repository*. [Online]. Available: http://archive.ics.uci.edu/ml

[93] V. W. Chandler and R. S. Lively, "2003 rock properties database: Density, magnetic susceptibility, and natural remanent magnetization of rocks in minnesota, version 2," Minnesota Geograph. Surv., Mounds View, MN, USA, Tech. Rep., 2010.

[94] S. Renjith, A. Sreekumar, and M. Jathavedan, "Evaluation of partitioning clustering algorithms for processing social media data in tourism domain," in *Proc. IEEE Recent Adv. Intell. Comput. Syst. (RAICS)*, Dec. 2018, pp. 127–131.

[95] James Yoo. *DBSAN: C++ implementation of DBSCAN Clustering Algorithm*. (Accessed: Jan. 16, 2021). [Online]. Available: https://github.com/james-yoo/DBSCAN

[96] A. A. Bushra. *Adilabdu/Revised-Dbscan*. (Accessed: Jan. 17, 2021). [Online]. Available: https://github.com/adilabdu/revised-dbscan

[97] CrikeeIP. *Optics-Clustering: An Algorithm for Finding Density-Based Clusters in Spatial Data*. (Accessed: Jan. 16, 2021). [Online]. Available: https://github.com/CrikeeIP/OPTICS-Clustering

[98] Ojmakhura. *HDBSCAN: This is an Implementation of the Hdbscan Algorithm Ricardo J.G.B. Campello*. (Accessed: Jan. 16, 2021). [Online]. Available: https://github.com/ojmakhura/hdbscan

[99] M. Goetz. *HPDBSCAN: Highly Parallel Dbscan (HPDBSCAN)*. (Accessed: Jan. 16, 2021). [Online]. Available: https://github.com/Markus-Goetz/hpdbscan

[100] M. Reber. (Nov. 2019). *SUBCLU Ruby Implementation*. [Online]. Available: https://code.michu-it.com/michael/programming-examples/src/branch/master/ruby/Algorithms/subclu.rb

[101] M. Folk, G. Heber, Q. Koziol, E. Pourmal, and D. Robinson, "An overview of the HDF5 technology suite and its applications," in *Proc. EDBT/ICDT Workshop Array Databases (AD)*, 2011, pp. 36–47.

[102] R. A. Brown, "Building a balanced k-d tree in O(kn log n) time," 2014, *arXiv:1410.5420*. [Online]. Available: http://arxiv.org/abs/1410.5420

[103] 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. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.

ADIL ABDU BUSHRA received the bachelor's degree in computer science from Addis Ababa University, Ethiopia, in 2019. He is currently pursuing the master's degree with the Department of Multimedia Engineering, Dongguk University, Seoul, South Korea. His research interests include machine learning, data science, and computational biology.

GANGMAN YI received the master's and Ph.D. degrees in computer science from Texas A&M University, USA, in 2007 and 2011, respectively. In 2011, he joined the System Software Group, Samsung Electronics, Suwon, South Korea. He was with the Department of Computer Science and Engineering, Gangneung-Wonju National University, South Korea, in 2012. Since 2016, he has been with the Department of Multimedia Engineering, Dongguk University, Seoul, South Korea. He has been researching in an interdisciplinary field of studies. His research interests include the development of computational methods to improve understanding of biological systems and its big data. He actively serves as a managing editor and a reviewer for international journals. He also serves as the chair for international conferences and workshops.