Comparative Evaluation of Region Query Strategies for DBSCAN Clustering*

Severino F. Galán
Artificial Intelligence Dept. at UNED
C/ Juan del Rosal, 16. 28040 Madrid, Spain.
E-mail: seve@dia.uned.es

Abstract

Clustering is a technique that allows data to be organized into groups of similar objects. DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) constitutes a popular clustering algorithm that relies on a density-based notion of cluster and is designed to discover clusters of arbitrary shape. The computational complexity of DBSCAN is dominated by the calculation of the ϵ-neighborhood for every object in the dataset. Thus, the efficiency of DBSCAN can be improved in two different ways: (1) by reducing the overall number of ϵ-neighborhood queries (also known as region queries), or (2) by reducing the complexity of the nearest neighbor search conducted for each region query. This paper deals with the first issue by considering the most relevant region query strategies for DBSCAN, all of them characterized by inspecting the neighborhoods of only a subset of the objects in the dataset. We comparatively evaluate these region query strategies (or DBSCAN variants) in terms of clustering effectiveness and efficiency; additionally, a novel region query strategy is introduced in this work. The results show that some specific DBSCAN variants are only slightly inferior to DBSCAN in terms of effectiveness, while greatly improving its efficiency. Among these variants, the novel one outperforms the rest.

Keywords: Clustering, DBSCAN algorithm, region query strategy, comparative evaluation.

1 Introduction

The real world generates an enormous amount of data about objects. Using this data to group objects into a set of categories is a frequent task in fields as diverse as artificial intelligence, biology, economy, or mathematics, among others. Clustering [20, 33, 2] is a technique that allows objects to be classified in such a manner that similar ones are grouped together. Although the concept of

cluster is subjective, given a representation of n objects, clustering algorithms usually obtain k groups according to a similarity measure. As a result, similar objects are assigned to the same group, whereas dissimilar ones are assigned to different groups.

Clustering algorithms can be divided into two main categories: hierarchical and partitional. *Hierarchical clustering* algorithms recursively partition the objects either in an agglomerative or in a divisive manner. In this way, a dendrogram is constructed which represents the nested grouping of objects. These algorithms have two drawbacks: (1) Due to their standard $\mathcal{O}(n^2)$ memory requirements and $\mathcal{O}(n^3)$ time complexity [47], the scalability offered is poor, and (2) Data objects incorrectly grouped at an early stage cannot be reallocated. Instead of creating a hierarchical structure for the clusters, *partitional clustering* algorithms generate the clusters as a single partition of the objects. The most popular partitional clustering algorithm is K-means [24], which has the following disadvantages: (1) It only works well on objects distributed as isotropic clusters, (2) It is sensitive to noisy objects and outliers, and (3) It requires the number of clusters to be specified in advance. The rest of the partitional clustering algorithms can be classified as follows:

1. **Model-based clustering** [25, 1] assumes that objects are generated from several probability distributions, such that objects in different clusters are generated by different probability distributions. The parameters of the mixture of probability models are usually established through the expectation-maximization algorithm [13].

2. **Search-based clustering** considers the search space formed by all the possible clustering configurations. In this search space, an optimization process is developed in order to reach a globally optimal clustering that fits the objects. Optimality is usually established through a function defined by applying a dissimilarity measure between cluster centers and objects. The search space of clustering configurations can be explored by using several search methods such as evolutionary algorithms [28, 11], tabu search [4, 9], simulated annealing [35, 32], or variable neighborhood search [18, 22], among others.

3. **Graph-based clustering** [46, 27] creates a graph representation of the objects by employing a similarity measure, and considers the interconnectivity and closeness of the graph nodes in order to obtain the clusters.

4. **Grid-based clustering** [17, 36] partitions the space into a number of cells, organized as a grid structure, where the clustering operations are performed. An advantage of grid-based clustering is that the processing time is significantly reduced.

5. **Density-based clustering** [21, 15, 40, 10] considers clusters as high density regions in the feature space, surrounded by low density regions. Density-based clustering algorithms search for connected dense regions, although the precise definition of connectedness varies depending on the specific algorithm.

Ester *et al.* [15] proposed a density-based clustering algorithm named DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) which has become one of the most popular and cited algorithms in the clustering literature [34, 14, 45, 5]. DBSCAN discovers clusters of arbitrary shape, handles noisy outliers effectively, and the number of clusters is not required in advance. In this algorithm, the first cluster is formed by searching the ϵ-neighborhood of an arbitrary initial object, incorporating the neighbors to the cluster, and repeating the search process for every neighbor provided

that their number is greater than a minimum threshold (in such a case, the neighbors are called "seeds"). When no further seeds are available for the current cluster, a new cluster is initiated from a non-visited randomly chosen object. In this way, two parameters are used by DBSCAN: on the one hand, the neighborhood size in terms of a distance ϵ and, on the other hand, the minimum number of objects in an ϵ-neighborhood that allows the neighbors to be considered as new seeds for the current cluster. In order to achieve more efficient ϵ-neighborhood searches for the region queries that take place in DBSCAN, an auxiliary R*-tree structure is constructed in advance from the whole dataset.

DBSCAN calculates the ϵ-neighborhood for each of the n objects in the dataset, thus ensuring that no cluster appears split after the execution of the algorithm. Since a region query takes $\mathcal{O}(\log n)$ time by using an R*-tree, the overall time complexity of DBSCAN is $\mathcal{O}(n \cdot \log n)$, as explained in [15, Section 4.1]. An immediate way of reducing this time complexity consists in generating the ϵ-neighborhoods of only a subset of the objects. This idea has been explored by several researchers who have developed a number of region query strategies (or DBSCAN variants) [41, 50, 8, 23] exhibiting a wide range of behaviors in terms of efficiency (computational complexity) and effectiveness (clustering quality). Although all of these variants are characterized by inspecting the ϵ-neighborhoods of only a subset of the objects in the dataset, each variant follows a specific strategy for generating cluster seeds.

The contribution of this paper is twofold. On the one hand, we comparatively evaluate a relevant set of the region query strategies suggested in the literature for DBSCAN clustering. On the other hand, we introduce a novel region query strategy for DBSCAN which is based on the best region query strategy resulting from the comparative evaluation. The experimental results show that some of the evaluated DBSCAN variants produce important advantages regarding efficiency and effectiveness. Furthermore, the novel variant introduced in this paper turns out to outperform the rest of the evaluated variants.

The rest of this paper is organized as follows. The DBSCAN clustering algorithm is described in Section 2. Section 3 analyzes DBSCAN and explains a set of relevant DBSCAN variants according to their region query strategy. In Section 4, the DBSCAN variants are experimentally evaluated. Finally, Section 5 concludes the paper with the main results derived from this work and several future research directions.

2 The DBSCAN Clustering Algorithm

DBSCAN [15] was proposed in the nineties as the first density-based clustering algorithm. Three important characteristics of DBSCAN are that clusters of arbitrary shape are discovered, noise is correctly identified, and the number of clusters is not required in advance. In DBSCAN, n region queries are required for a dataset containing n objects, where each region query consists of calculating the local neighbors of a specific object in the dataset. The extension of the cluster containing the current object (or the definition of a new cluster from the current object) is subject to the condition that the neighborhood around the current object contains at least a given number of objects. A spatial access data structure which is very efficient for points, called R*-tree [7], is used to obtain the local neighborhood of an object in $\mathcal{O}(\log n)$ time. Thus, as described in [15, Section 4.1], the overall computational complexity of DBSCAN is $\mathcal{O}(n \cdot \log n)$. It is important to note that the R*-tree must be built before clustering the dataset.

Given a dataset D of n objects, a distance function δ defined over $D \times D$, and two input parameters

$\epsilon \in \mathbb{R}$ and $MinPts \in \mathbb{N}$ (minimum number of points), DBSCAN relies on the following definitions:

Definition 1 (**ϵ-neighborhood**). *The ϵ-neighborhood of an object $o \in D$, denoted as $N_\epsilon(o)$, is defined by $N_\epsilon(o) = \{o' \in D \mid \delta(o, o') \leq \epsilon\}$.*

Definition 2 (**directly density-reachable**). *An object $o' \in D$ is directly density-reachable from an object $o \in D$ if $|N_\epsilon(o)| \geq MinPts$ and $o' \in N_\epsilon(o)$. (The first condition establishes that o is a core point.)*

Definition 3 (**density-reachable**). *An object $o' \in D$ is density-reachable from an object $o \in D$ if there is a sequence of z objects $\{o_1, \ldots, o_z\}$, with $o_1 = o$, $o_z = o'$, and $z \in \{2, \ldots, n\}$, such that $o_{i+1} \in D$ is directly density-reachable from $o_i \in D$ for $i \in \{1, \ldots, z-1\}$.*

Definition 4 (**density-connected**). *An object $o_1 \in D$ is density-connected to an object $o_2 \in D$ if there is an object $o \in D$ such that both o_1 and o_2 are density-reachable from o.*

Definition 5 (**cluster**). *A density-based cluster C is a non-empty subset of D such that:*
(1) $\forall o \in C$ and $o' \in D$, if o' is density-reachable from o then $o' \in C$ (maximality property), and
(2) $\forall o_1, o_2 \in C$, o_1 is density-connected to o_2 (connectivity property).

Definition 6 (**noise**). *Let $\{C_1, \ldots, C_k\}$ be the clusters in D, with $k \in \{0, \ldots, n/MinPts\}$. The noise N is the set of objects in D not belonging to any cluster, that is, $N = D \setminus \{C_1, \ldots, C_k\}$.*

DBSCAN starts by performing a region query for an arbitrary object o in order to calculate its ϵ-neighborhood. If $|N_\epsilon(o)| < MinPts$, the object o is temporarily labeled as noise; otherwise, the object o and its neighbors are labeled to belong to cluster C_1, and the region query is repeated for each of o's neighbors. When the current cluster C_1 cannot be expanded further, an arbitrary unlabeled object is selected and a new cluster C_2 is generated from the selected object. This operation is iterated for clusters $\{C_i\}$ until all of the objects have been labeled.

Since a region query is carried out in DBSCAN for every object, an excessive amount of time may be spent in dense regions of the dataset. The unlabeled objects in the ϵ-neighborhood of a core object are usually called "seeds" and, during the early stages of the clustering process, the number of seed objects for a cluster increases monotonously in such a way that its maximum size is unpredictable. The next section explains the most relevant region query strategies for DBSCAN that can be found in the literature. Other methods that improve or extend DBSCAN's capabilities in different ways are the following:

1. The time complexity of DBSCAN is lowered in [44, 3, 12] by combining DBSCAN with grid-based clustering techniques.

2. The problem of detecting clusters in datasets with varying densities is addressed in [5, 29, 48] in the context of DBSCAN clustering.

3. The typical crisp neighborhood used in DBSCAN is extended in [39, 26] by considering the more general fuzzy neighborhood.

4. The clustering accuracy of DBSCAN is improved in [19] by adjusting the parameters *MinPts* and ϵ with the help of a genetic algorithm.

5. Parallel versions of DBSCAN are described in [45, 37].

6. DBSCAN is enhanced in [14, 6] in order to perform incremental clustering in dynamic databases.

3 Region Query Strategies for DBSCAN

DBSCAN works by calculating the ϵ-neighborhood of every object in the dataset. Overall, such region queries constitute the most expensive operation in DBSCAN and determine its computational complexity. Several DBSCAN variants (see citations in Section 3.2) have been designed in order to reduce the number of ϵ-neighborhood queries, while trying to preserve the quality of the resulting clusters as much as possible. For that purpose, the number of region queries is decreased by heuristically selecting some representative objects as seeds. Note that, in the worst case, the time complexity of any DBSCAN variant is the same as that of DBSCAN. This is due to the fact that, given a DBSCAN variant, it is always possible to design a problem for which the variant needs to calculate the ϵ-neighborhoods of all the objects. Nonetheless, for real-world problems, the DBSCAN variants will typically generate time savings derived from a reduction in the number of ϵ-neighborhood queries. The specific magnitude of the reduction will depend on the particular problem and DBSCAN variant considered.

3.1 Analysis of DBSCAN

Two extreme cases are worth analyzing when the neighbor y of a core object x is considered for expansion under DBSCAN:

- First, if y's neighborhood is fully covered by the neighborhoods of other x's neighbors, the region query for y can be omitted. In other words, the clustering expansion from x will produce the same results regardless of whether y is selected as a seed. This case frequently arises in dense clusters, where many objects can be ignored for expansion purposes. For instance, Figure 1(a) depicts a core object x and five of its neighbors $\{a, b, c, d, y\}$. As shown in Figure 1(b), y's neighborhood (thick line circle) is covered by the union of the neighborhoods of $\{a, b, c, d\}$ (thin line circles). (Note that we are assuming Euclidean distance in Figure 1.) Consequently, once objects $\{a, b, c, d\}$ have been included as seeds during x's cluster expansion, object y can be ignored.

- Second, if y is not included as a seed during the clustering expansion from x, it may happen that x's cluster is eventually split. Although this second case is much less frequent than the first case explained above, it produces the undesired effect of poorer clustering quality. Sparse clusters are more likely to suffer from this behavior. For instance, Figure 2 depicts a core object x_1 and its seven neighbors, along with a core object x_2 and its five neighbors; as shown in the figure, object y_1 belongs to x_1's neighborhood and object y_2 belongs to x_2's neighborhood. Suppose that x_1 is first expanded and all of its neighbors are selected as seeds except y_1. Then, suppose that x_2 is expanded and object y_2 is ignored as well. As a result, if we assume that $MinPts < 3$, the cluster formed by the fourteen objects drawn in Figure 2 is unexpectedly split in two parts: one of them contains x_1, y_1, and the six left-hand objects, whereas the other contains x_2, y_2, and the four right-hand objects.

The ideal region query strategy for DBSCAN would be to expand the minimum number of seeds that guarantee the preservation of all clusters. This strategy would produce maximum efficiency without reducing effectiveness. However, since this is difficult to achieve in practice, heuristic methods have been designed that approximate the ideal strategy. These heuristic strategies do not guarantee

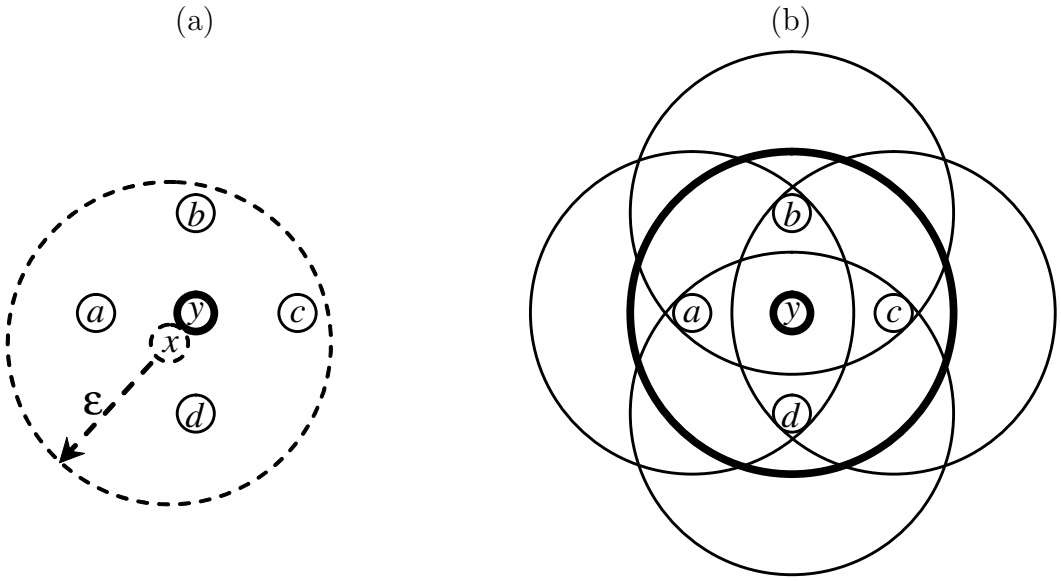


Figure 1: A core object x and five of its ϵ-neighbors $\{a, b, c, d, y\}$ are shown in part (a) of the figure. Part (b) shows how the neighborhood of one of x's neighbors, y, is fully covered by the neighborhoods of other x's neighbors, $\{a, b, c, d\}$. Thus, object y can be omitted in the cluster expansion from x.

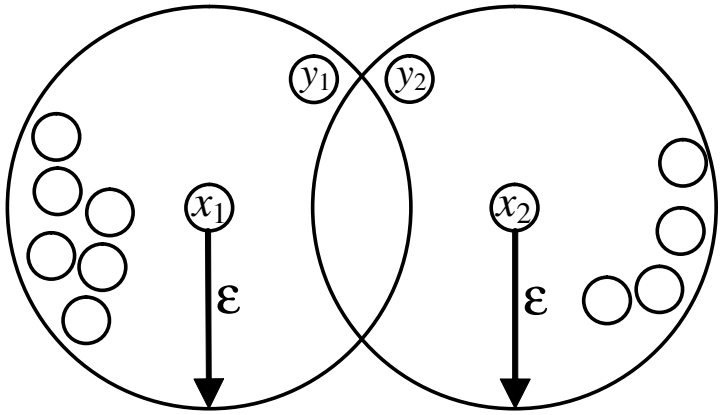


Figure 2: A cluster formed by fourteen objects for $MinPts < 3$: two core objects, x_1 and x_2, and their neighbors. When objects y_1 and y_2 are ignored as seeds, the cluster is split. Consequently, x_1 and x_2 become disconnected.

the final clustering quality obtained by DBSCAN and, as a result, they may lead to some DBSCAN clusters being split.

3.2 DBSCAN Variants

The present section describes the most relevant DBSCAN variants found in the literature in terms of the region query strategy used. These variants, which are empirically evaluated in an extensive way in Section 4, are the following:

1. **DBRS:**
   The *Density-Based spatial clustering method with Random Sampling* (DBRS) [41, 42] repeatedly chooses an unlabeled object at random and examines its neighborhood. If the neighborhood is sparsely populated, the object is labeled as noise; otherwise, if any particular object in the neighborhood is part of a known cluster, the neighborhood is joined to that cluster. If neither of these two possibilities applies, a new cluster is created with the neighborhood. The procedure is iterated until every object in the dataset is clustered or classified as noise. Therefore, the region query strategy developed by DBRS is quite different from that of DBSCAN. This is due to the fact that, when the current object is expanded and the objects in its neighborhood are clustered, none of them is allowed to become a seed for future expansions. This constitutes an important disadvantage for DBRS, since some of the clusters created by DBSCAN might be unexpectedly split.

2. **FDBSCAN#1:**
   We denote as FDBSCAN#1 the fast density-based clustering algorithm based on DBSCAN that was introduced in [23]. Like DBRS, FDBSCAN#1 allows seeds to be selected outside the current ϵ-neighborhood. The only difference with DBRS is that the objects are initially sorted by a certain coordinate (for example, the x-coordinate in two-dimensional space). The next unlabeled ordered object is selected as the next seed. Despite the fact that the behavior of DBRS is improved as a consequence of ordering the objects, FDBSCAN#1 remains quite inferior to DBSCAN in terms of effectiveness.

3. **DBSCAN*max*:**
   We introduce in this paper a DBSCAN variant that, rather than expanding all of the objects in the ϵ-neighborhood of a core object x, only expands one of them. We name the algorithm DBSCAN*max*, since the object in $N_\epsilon(x)$ selected as a seed is the one that maximizes the distance to x. In this way, contrarily to DBRS and FDBSCAN#1, DBSCAN*max* selects seeds exclusively from those inside the current ϵ-neighborhood. Although this constitutes an advantage in terms of effectiveness, the potential of the method is reduced by the fact that only one seed is generated for each neighborhood. The DBSCAN variants considered in the rest of this section follow a similar pattern to that of DBSCAN*max*, although they expand more than one object in the ϵ-neighborhood of a core object. DBSCAN*max* is introduced in this paper in order to fill a conceptual gap present in the literature and provide an exhaustive range of representative region query strategies for DBSCAN. However, as shown in Section 4, DBSCAN*max* turns out not to be a competitive variant. Nonetheless, another novel DBSCAN variant that outperforms the rest of the variants (see Section 4) is introduced later in the present section.

4. **FDBSCAN#2:**
   We denote as FDBSCAN#2 the fast density-based clustering algorithm based on DBSCAN that was proposed in [50, 49]. FDBSCAN#2 is based on the idea that the outer objects in the ϵ-neighborhood of a core object are favorable seed candidates, since the ϵ-neighborhoods of inner objects tend to be covered by those of outer objects. Specifically, for two-dimensional data, four objects are selected as seeds in the ϵ-neighborhood of a core object: the leftmost, the rightmost, the uppermost, and the lowermost objects. The reason is that such an ϵ-neighborhood can be approximately covered by four well scattered seeds. This approach allows FDBSCAN#2 to obtain results close to those of DBSCAN, although there is still room for improvement regarding effectiveness. In general, for d-dimensional data, $2 \times d$ seeds are selected for cluster expansion from the ϵ-neighborhood of each core object, that is, two seeds are selected over each dimension.

5. **IDBSCAN:**
   The *Improved sampling-based DBSCAN algorithm* (IDBSCAN) [8] marks, as shown in Figure 3 for the case of two-dimensional space, eight points in the boundary of the current ϵ-neighborhood, $N_\epsilon(x)$. If the origin of coordinates is assumed to be located at object x, the eight marked points are those with coordinates:

   $$\{(\epsilon \cdot \cos(\alpha), \epsilon \cdot \sin(\alpha))\} \text{ for } \alpha \in \{90°, 45°, 0°, -45°, -90°, -135°, 180°, 135°\},$$

   where α is the angle of the marked point with respect to the horizontal axis. IDBSCAN operates such that, for each of the eight marked points, its nearest object in $N_\epsilon(x)$ is selected as a seed. (Duplicate seeds are ignored.) Usually, this method gives rise to a similar behavior to that of DBSCAN. Although improving IDBSCAN's effectiveness is a challenging task, an important contribution of the present work is to demonstrate how the robustness of IDBSCAN can be increased. In general, for the d-dimensional case, $3^d - 1$ marked points are considered.

6. **I*DBSCAN:**
   The present work introduces a novel DBSCAN variant called I*DBSCAN, which is a generalization of IDBSCAN. Under I*DBSCAN, the eight marked points in two dimensions are the following (see Figure 4):

   $$\{(\epsilon \cdot \cos(\alpha + \alpha'), \epsilon \cdot \sin(\alpha + \alpha'))\} \text{ for } \alpha \in \{90°, 45°, 0°, -45°, -90°, -135°, 180°, 135°\},$$

   where α' is an angle that is randomly chosen within the interval $[0°, 45°]$. Every time a new region query is carried out by I*DBSCAN, a new angle α' is randomly generated. Unlike IDBSCAN, I*DBSCAN is based on the idea that no specific directions are preferred in the clustering process and, as a consequence, a random component is introduced. This idea represents the main advantage of I*DBSCAN over IDBSCAN. In other words, given that there are no preferred directions in which the objects to be clustered are spatially distributed, the seeds should be generated isotropically and not following fixed directions. Section 4 shows that I*DBSCAN outperforms the rest of the variants explained in the present section.

4 Experimental Evaluation

In this section, we compare DBSCAN with the six variants explained in Section 3.2 in terms of clustering quality and running time. The results obtained for DBSCAN constitute the ideal clustering

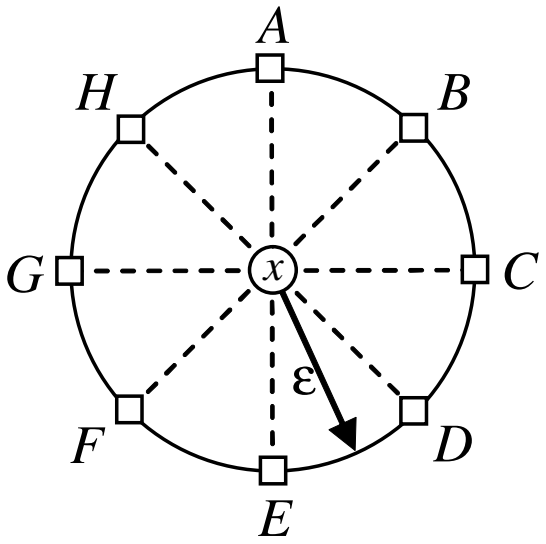


Figure 3: The eight marked boundary points $\{A, B, C, D, E, F, G, H\}$ used by the IDBSCAN algorithm in the case of two-dimensional space.

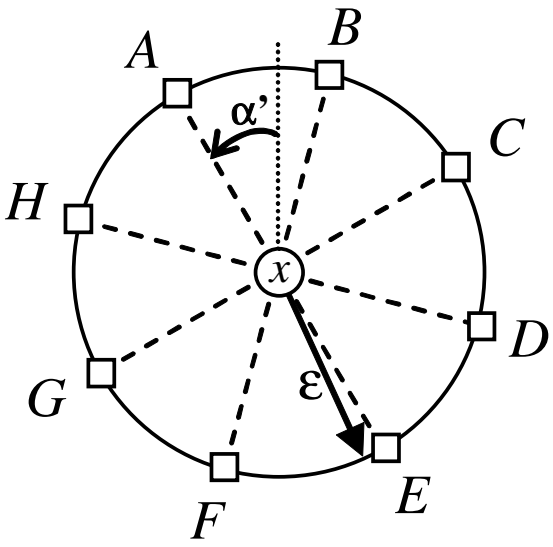


Figure 4: The eight marked boundary points $\{A, B, C, D, E, F, G, H\}$ used by the I*DBSCAN algorithm in the case of two-dimensional space. The angle α' is randomly chosen within the interval $[0°, 45°]$ every time a new region query is carried out.

quality, which cannot be improved by its variants. In general, the six variants take less running time than DBSCAN at the expense of producing an undesired reduction in the ideal clustering quality. This quality reduction varies significantly among the variants. There are some dataset instances for which some variants obtain poor quality results compared to DBSCAN, whereas other variants reach even the same quality as that obtained by DBSCAN.

4.1 Experimental Setup

We implement DBSCAN and its tested variants within NetLogo [43], an agent-based modeling and programming environment. The experiments are carried out on a computer with a 2.67 GHz processor, 8Gb of memory, and running Windows. The evaluation is performed in the following context:

- The considered dataset instances are defined in a two-dimensional space.
- DBSCAN and its variants are applied with $\epsilon = 1$. Note that every clustering problem P to which DBSCAN is applied with $\epsilon \neq 1$, can be transformed into an equivalent problem P' with $\epsilon' = 1$ by scaling up or down the objects' coordinates in P by a factor of $1/\epsilon$. By using $\epsilon = 1$, we take advantage of the NetLogo architecture in order to carry out efficient grid-based region queries.[1]
- We assume that every object is a core object (see Definition 2) regardless of the number of objects contained in its ϵ-neighborhood. In this way, each object belongs to one of the clusters, and the noise set is empty. If we consider that $\forall o \in D$, $o \in N_\epsilon(o)$, this assumption is equivalent to setting $MinPts = 1$.

We employ both artificially generated and real-world datasets in the experiments. The coordinates (x, y) of every object in the artificially generated datasets satisfy the conditions: $0 \leq x \leq 100$ and $0 \leq y \leq 50$. The tested datasets can be classified into the following groups, where the first five groups correspond to synthetic datasets and the last two groups correspond to real-world datasets:

1. **Random points** rPoints$N$:
   Dataset with N objects randomly distributed with uniform probability over the drawing frame. An instance of rPoints2000 is depicted in Figure 5(a).
2. **Random line segments** rSegments$N$:
   Dataset with N objects arranged as line segments, where each segment consists of a set of objects separated by a step distance of 0.5. The number of objects for each segment is a random integer from the range $\{1, \ldots, 50\}$. The position for each segment is randomly chosen with uniform probability distribution within the drawing frame, and its direction (angle with respect to the vertical axis) is randomly chosen with uniform probability distribution over the real interval $[0, 360)$. Figure 5(b) contains an instance of rSegments2000.

[1] Although NetLogo provides four types of agents: turtles, patches, links, and the observer, we only make use of turtles and patches in our clustering implementation. Specifically, objects are represented as turtles located in a two-dimensional world that is divided up into a grid of patches. Each patch is a 1x1 square piece of world that has direct access to the set of turtles located in the patch. As explained in [34, Section 5.1], we take advantage of this fact in order to implement grid-based ϵ-neighborhood queries that take $\mathcal{O}(1)$ time, thus speeding up the $\mathcal{O}(\log n)$ time spent for each region query by using spatial indices like the R*-tree.

3. **Random stars** rStars$N$:
   Dataset with N objects arranged as stars, where each star consists of five segments of the same length. The five segments of a star have one end in common and the following angles with respect to the vertical axis: $\{360 \cdot i/5\}$ for $i \in \{0, 1, 2, 3, 4\}$. The number of objects for each segment of a star is a random integer from the range $\{1, \ldots, 25\}$. The position for each star is randomly chosen with uniform probability distribution within the drawing frame. Figure 5(c) shows an instace of rStars2000.

4. **Random bubbles** rBubbles$N.S.L$:
   Dataset with N objects arranged as bubbles, where each bubble is a set of objects located inside a small circle. We implement bubbles as stars with a high number S of line segments. Each segment of a bubble typically contains a small number of objects, defined by a random integer from the range $\{1, \ldots, L\}$. Figure 5(d) contains an instance of rBubbles40960.32.6.

5. **Arbitrary shapes** Arbitrary$N$:
   Dataset with N objects arranged as arbitrary shapes. Figure 6 shows the four arbitrary datasets used in our experiments.

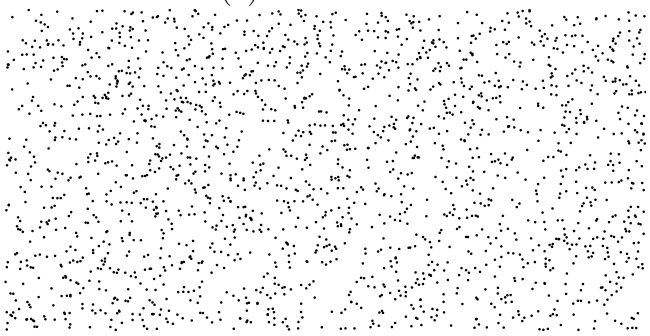
6. **SEQUOIA 2000**:
   The publicly available SEQUOIA 2000 benchmark database [38] uses real datasets from Earth Science grouped in four categories: raster data, point data, polygon data, and directed graph data. The geographic point dataset in SEQUOIA 2000 contains 62556 Californian landmarks and their location coordinates. In our work, these locations are included in a 200x250 drawing frame (see Figure 7(a)).

7. **FINLAND**:
   Dataset containing the locations of 13467 MOPSI users in Finland as of 2012.[2] These locations are included in a 100x215 drawing frame (see Figure 7(b)).

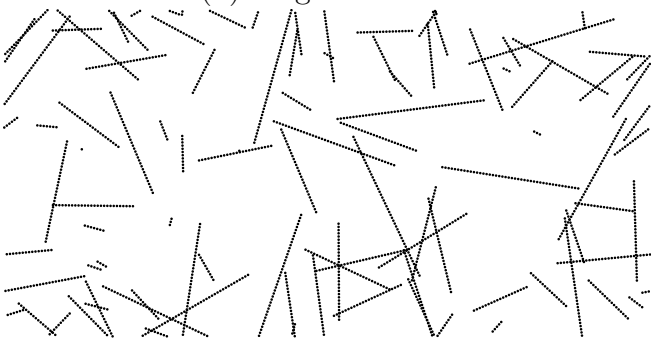
In order to determine the effectiveness (or clustering quality) of the tested algorithms, we measure the *mean number of clusters* produced by each algorithm when applied to each dataset instance over a fixed number of runs. In the same way, the *mean running times* are measured in order to establish the efficiency of each algorithm. We additionally obtain two useful measures: on the one hand, the *mean percentage of seeds* (mean percentage of objects in the dataset from which a region query is carried out) and, on the other hand, the *mean number of cluster updates* (when two different clusters are merged, a cluster update consists in changing the cluster label of all the objects in one of the clusters). In order to implement the cluster updating operations, we use a tree to connect the objects belonging to the same cluster. These trees are constructed such that, every time a region query operation is carried out, the expanded seed is connected with only two types of neighbors: (1) those which are unlabeled, and (2) those belonging to a different cluster than that of the expanded seed. In the latter case, the objects of one of the two merged trees need to be relabeled. Note that this can only happen with the DBSCAN variants, but not with DBSCAN itself, where the expanded seed and its neighbors labeled in previous steps always belong to the same cluster. Consequently, no cluster update takes place in DBSCAN.

[2]Publicly available at http://cs.joensuu.fi/sipu/datasets/

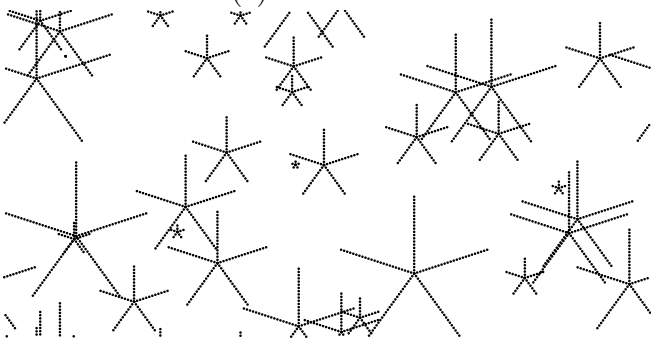
(a) rPoints2000



(b) rSegments2000



(c) rStars2000



(d) rBubbles40960.32.6

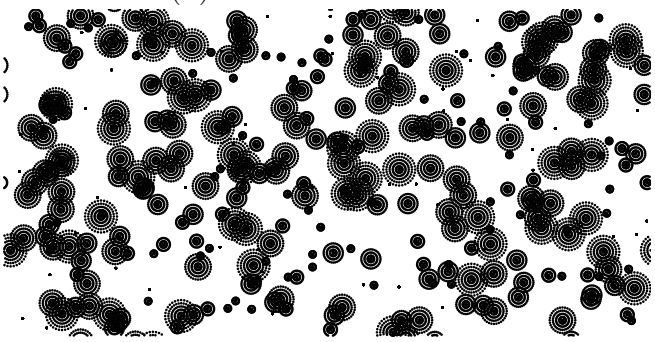
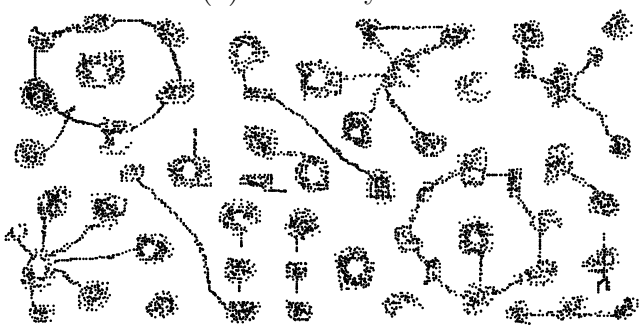
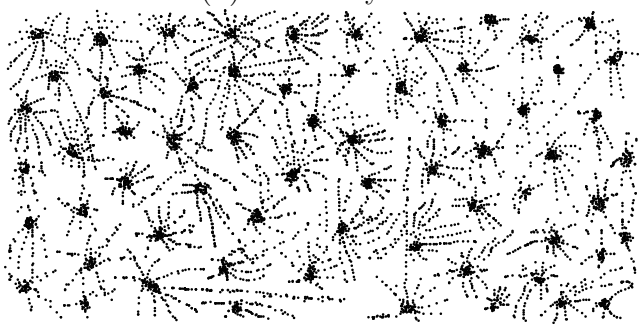


Figure 5: Examples of the first four groups of artificially generated datasets used in the experiments.

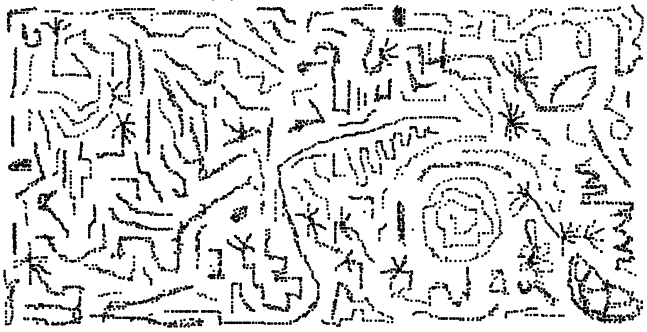
(a) Arbitrary7000



(b) Arbitrary8000



(c) Arbitrary9000



(d) Arbitrary10000

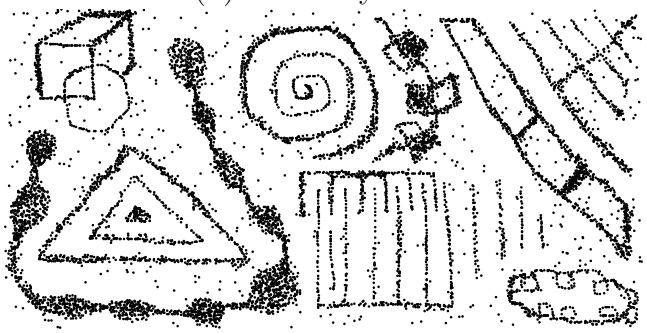


Figure 6: The four arbitrary datasets used in the experiments.

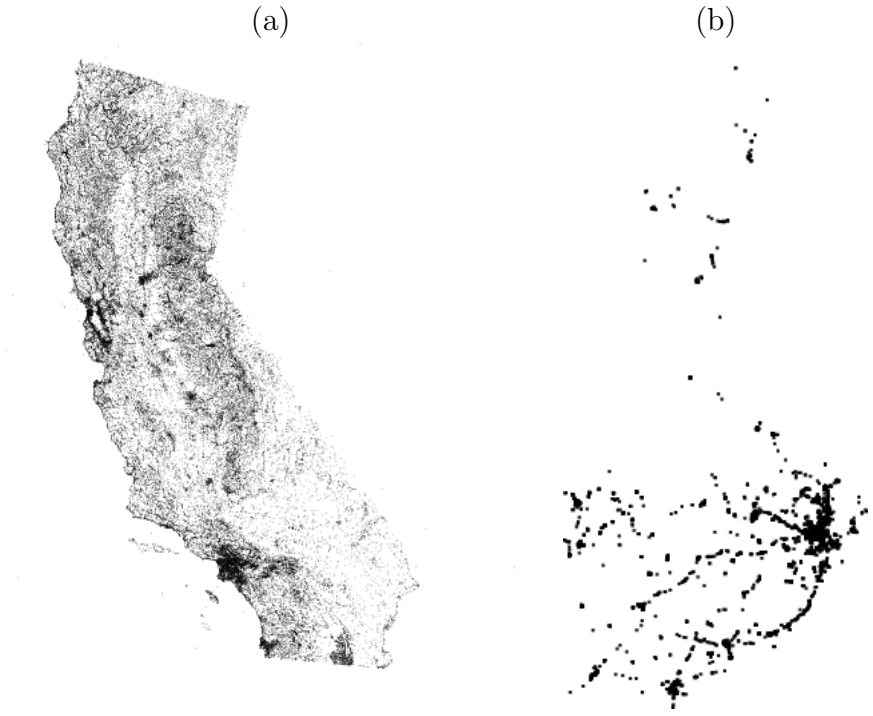


Figure 7: The SEQUOIA 2000 (a) and the FINLAND (b) real-world datasets used in the experiments.

4.2 Experimental Results

DBSCAN and the six variants explained in Section 3.2 are executed on the following datasets:

- An instance of rPoints$N$, rSegments$N$, and rStars$N$ for $N \in \{2000, 10000, 50000, 250000\}$.
- An instance of rBubbles$N.S.L$ for $N.S.L \in \{40960.32.6, 54272.64.4, 80000.32.3, 100032.64.4, 200000.64.2, 300000.128.3, 400000.128.2, 500096.128.2\}$.
- An instance of Arbitrary$N$ for $N \in \{7000, 8000, 9000, 10000\}$.
- The geographic point dataset from California in SEQUOIA 2000.
- The FINLAND dataset containing the MOPSI users' locations.

These dataset instances cover a wide range of cases in terms of number of objects and their spatial distribution. Tables 1 through 4 contain the numerical results obtained for the mean number of clusters, the mean running times, the mean percentage of seeds, and the mean number of cluster updates, respectively. The results in the four tables are averaged over one hundred runs.

Before discussing the experimental results obtained in Tables 1-4, their statistical significance needs to be established. In other words, for each of the four tables, it needs to be determined whether or not the data (grouped by column) come from the same distribution and, therefore, the differences are due to random effects (null hypothesis). Table 5 contains the results for ANOVA tests [16] on each of the four tables. Besides the F value, the probability p of the corresponding result assuming the null hypothesis is shown. When probability p is less than a small threshold (typically, $p = 0.05$ is

Dataset Instance	*DBSCAN*	*I*DBSCAN*	*IDBSCAN*	*FDBSCAN#2*	*DBSCANmax*	*FDBSCAN#1*	*DBRS*
rPoints2000	**1038.00**	**1038.00**	**1038.00**	**1038.00**	1039.16	1059.00	1066.44
rPoints10000	**39.00**	**39.00**	**39.00**	**39.00**	51.76	86.00	200.70
rPoints50000	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	1.09
rPoints250000	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
rSegments2000	**29.00**	**29.00**	**29.00**	**29.00**	36.42	38.00	157.62
rSegments10000	**4.00**	**4.00**	**4.00**	**4.00**	5.74	9.00	79.83
rSegments50000	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	1.02
rSegments250000	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
rStars2000	**11.00**	**11.00**	**11.00**	**11.00**	20.48	28.26	122.30
rStars10000	**1.00**	**1.00**	**1.00**	**1.00**	4.78	13.44	90.90
rStars50000	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	1.46
rStars250000	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
rBubbles40960.32.6	**63.00**	63.09	63.16	63.33	68.03	67.58	72.14
rBubbles54272.64.4	**149.00**	149.50	149.60	150.64	154.22	152.53	157.33
rBubbles80000.32.3	**186.00**	186.47	186.53	188.52	212.85	203.42	232.33
rBubbles100032.64.4	**151.00**	151.53	151.70	152.65	160.76	155.71	167.34
rBubbles200000.64.2	**318.00**	321.21	322.01	331.08	372.72	383.54	442.67
rBubbles300000.128.3	**182.00**	182.59	182.79	185.53	207.34	205.68	227.54
rBubbles400000.128.2	**300.00**	301.56	302.80	309.25	357.58	366.45	430.70
rBubbles500096.128.2	**239.00**	241.80	241.76	249.14	303.17	302.49	385.37
Arbitrary7000	**26.00**	**26.00**	**26.00**	**26.00**	28.08	33.97	63.63
Arbitrary8000	**366.00**	**366.00**	**366.00**	**366.00**	388.99	416.87	563.33
Arbitrary9000	**130.00**	**130.00**	**130.00**	130.05	141.48	178.81	308.24
Arbitrary10000	**224.00**	**224.00**	**224.00**	**224.00**	229.80	254.50	301.16
SEQUOIA 2000	**652.00**	**652.00**	**652.00**	**652.00**	677.67	723.00	864.30
FINLAND	**267.00**	**267.00**	**267.00**	**267.00**	267.72	274.50	287.38

Table 1: Mean number of clusters, averaged over one hundred runs, for the seven clustering algorithms tested in the experiments. The best result for each row is highlighted in bold.

Dataset Instance	*DBSCAN*	*I*DBSCAN*	*IDBSCAN*	*FDBSCAN#2*	*DBSCANmax*	*FDBSCAN#1*	*DBRS*
rPoints2000	0.080	0.098	0.085	0.085	0.089	**0.015**	0.092
rPoints10000	0.152	0.209	0.191	0.171	0.448	**0.106**	0.982
rPoints50000	1.786	1.294	1.225	0.862	3.289	**0.664**	17.718
rPoints250000	41.394	8.536	8.030	5.341	17.044	**4.499**	90.450
rSegments2000	0.025	0.031	0.031	0.025	0.027	**0.017**	0.052
rSegments10000	0.140	0.187	0.180	0.144	0.266	**0.094**	1.262
rSegments50000	1.858	1.243	1.192	0.820	3.141	**0.527**	16.352
rSegments250000	38.473	7.953	7.652	5.033	15.809	**4.217**	89.185
rStars2000	0.021	0.033	0.030	0.024	0.029	**0.016**	0.046
rStars10000	0.151	0.180	0.169	0.134	0.439	**0.088**	1.214
rStars50000	1.955	1.221	1.183	0.798	3.106	**0.571**	16.179
rStars250000	41.990	7.896	7.576	4.934	15.203	**4.242**	92.198
rBubbles40960.32.6	2.632	1.064	1.094	0.880	1.949	**0.418**	7.014
rBubbles54272.64.4	6.115	1.752	1.723	1.588	2.056	**0.522**	5.996
rBubbles80000.32.3	8.027	3.013	2.963	2.712	5.537	**0.843**	14.035
rBubbles100032.64.4	13.688	3.416	3.419	2.975	5.249	**1.075**	16.211
rBubbles200000.64.2	39.819	10.218	10.014	9.679	15.910	**2.368**	29.991
rBubbles300000.128.3	93.066	12.244	11.924	10.637	22.506	**4.090**	55.263
rBubbles400000.128.2	141.073	20.845	20.942	18.002	33.116	**5.734**	61.690
rBubbles500096.128.2	203.475	23.148	24.858	22.064	47.074	**8.017**	99.601
Arbitrary7000	0.130	0.132	0.125	0.093	0.105	**0.057**	0.254
Arbitrary8000	0.361	0.322	0.317	0.283	0.307	**0.065**	0.474
Arbitrary9000	0.204	0.190	0.183	0.148	0.251	**0.075**	0.463
Arbitrary10000	0.415	0.391	0.388	0.333	0.297	**0.089**	0.536
SEQUOIA 2000	7.274	6.288	6.563	6.230	17.706	**0.906**	49.218
FINLAND	38.543	0.490	0.504	0.395	0.336	**0.131**	0.325

Table 2: Mean running times in seconds, averaged over one hundred runs, for the seven clustering algorithms tested in the experiments. The results are rounded to three decimal places. The best result for each row is highlighted in bold.

Dataset Instance	*DBSCAN*	*I*DBSCAN*	*IDBSCAN*	*FDBSCAN#2*	*DBSCANmax*	*FDBSCAN#1*	*DBRS*
rPoints2000	100.00	99.86	99.81	98.83	87.22	63.70	**62.05**
rPoints10000	100.00	92.50	92.15	80.37	38.16	26.96	**23.71**
rPoints50000	100.00	47.10	47.92	30.33	8.40	7.67	**6.02**
rPoints250000	100.00	12.62	12.86	7.03	1.73	1.83	**1.32**
rSegments2000	100.00	98.37	97.62	93.24	57.63	36.60	**31.52**
rSegments10000	100.00	86.46	86.32	71.83	30.78	23.27	**19.59**
rSegments50000	100.00	46.40	47.27	29.95	8.31	7.76	**6.08**
rSegments250000	100.00	12.77	13.03	7.10	1.74	1.84	**1.34**
rStars2000	100.00	90.87	87.29	85.49	50.42	32.44	**28.26**
rStars10000	100.00	81.72	80.73	68.07	29.81	21.80	**18.52**
rStars50000	100.00	45.24	46.05	29.33	8.28	7.66	**5.99**
rStars250000	100.00	12.72	12.98	7.07	1.74	1.84	**1.33**
rBubbles40960.32.6	100.00	30.61	30.90	20.46	6.08	5.36	**4.33**
rBubbles54272.64.4	100.00	14.14	13.93	8.75	2.82	2.23	**1.89**
rBubbles80000.32.3	100.00	16.81	16.50	10.32	3.24	2.55	**2.17**
rBubbles100032.64.4	100.00	12.49	12.34	7.57	2.33	1.91	**1.60**
rBubbles200000.64.2	100.00	7.09	6.93	3.93	1.27	0.85	**0.77**
rBubbles300000.128.3	100.00	4.97	4.84	2.85	0.89	0.70	**0.60**
rBubbles400000.128.2	100.00	3.62	3.54	1.97	0.63	0.42	**0.38**
rBubbles500096.128.2	100.00	3.41	3.33	1.84	0.56	0.40	**0.35**
Arbitrary7000	100.00	72.11	71.51	55.74	20.62	15.62	**13.23**
Arbitrary8000	100.00	63.83	63.53	53.93	30.67	20.54	**18.46**
Arbitrary9000	100.00	75.93	73.24	62.56	25.20	16.88	**14.92**
Arbitrary10000	100.00	64.63	63.62	50.34	18.87	14.35	**12.54**
SEQUOIA 2000	100.00	58.61	58.75	43.67	16.73	13.06	**11.09**
FINLAND	100.00	12.54	12.49	9.91	5.29	3.50	**3.27**

Table 3: Mean percentage of seeds, averaged over one hundred runs, for the seven clustering algorithms tested in the experiments. The results are rounded to two decimal places. The best result for each row is highlighted in bold.

Dataset Instance	*DBSCAN*	*I*DBSCAN*	*IDBSCAN*	*FDBSCAN#2*	*DBSCANmax*	*FDBSCAN#1*	*DBRS*
rPoints2000	**0.00**	**0.00**	0.24	0.72	62.28	280.65	225.52
rPoints10000	**0.00**	1.94	4.24	22.56	21555.77	8939.90	24950.47
rPoints50000	**0.00**	**0.00**	**0.00**	**0.00**	38640.14	7469.87	124655.28
rPoints250000	**0.00**	**0.00**	**0.00**	**0.00**	194387.61	11120.50	552093.70
rSegments2000	**0.00**	**0.00**	2.82	3.36	1874.52	2039.07	1812.20
rSegments10000	**0.00**	**0.00**	0.07	1.99	9739.53	5159.49	29582.42
rSegments50000	**0.00**	**0.00**	**0.00**	**0.00**	12527.91	8024.09	125683.52
rSegments250000	**0.00**	**0.00**	**0.00**	**0.00**	48114.40	10909.23	514054.41
rStars2000	**0.00**	**0.00**	27.33	22.18	1949.66	1937.76	1957.29
rStars10000	**0.00**	**0.00**	2.72	3.05	11761.12	5450.92	29037.54
rStars50000	**0.00**	**0.00**	**0.00**	0.04	18091.66	8367.18	123372.11
rStars250000	**0.00**	**0.00**	**0.00**	**0.00**	82731.19	13009.86	524908.57
rBubbles40960.32.6	**0.00**	93.70	128.27	329.45	30570.30	15650.86	61744.15
rBubbles54272.64.4	**0.00**	325.06	335.80	425.47	10108.99	8582.55	27926.25
rBubbles80000.32.3	**0.00**	1397.47	2068.12	2593.11	45174.94	24429.57	74403.14
rBubbles100032.64.4	**0.00**	335.85	785.06	1931.45	40945.31	21621.11	87651.78
rBubbles200000.64.2	**0.00**	6548.10	9108.93	8289.04	76855.40	43395.81	102067.92
rBubbles300000.128.3	**0.00**	4570.83	8224.58	13274.05	145853.10	81910.59	271516.25
rBubbles400000.128.2	**0.00**	9592.95	17895.21	22537.37	159228.34	104521.53	208016.35
rBubbles500096.128.2	**0.00**	27207.37	34202.83	36169.48	294104.79	118769.16	365586.17
Arbitrary7000	**0.00**	0.1	9.51	2.90	5137.84	3416.82	7851.99
Arbitrary8000	**0.00**	2.51	149.91	79.75	3040.09	3765.26	3749.28
Arbitrary9000	**0.00**	**0.00**	63.49	35.69	3931.72	4675.18	6658.60
Arbitrary10000	**0.00**	1.17	19.98	49.22	9555.99	5556.24	13994.60
SEQUOIA 2000	**0.00**	7.00	18.77	64.98	96570.88	22471.99	186450.45
FINLAND	**0.00**	3.26	2.86	5.98	2455.19	1473.97	2661.92

Table 4: Mean number of cluster updates, averaged over one hundred runs, for the seven clustering algorithms tested in the experiments. The best result for each row is highlighted in bold.

Table	F	F_{critical}	p
1	0.292	2.151	0.94
2	5.432	2.151	3.56E-5
3	40.728	2.151	8.41E-31
4	12.124	2.151	2.2266E-11

Table 5: ANOVA tests for Tables 1 through 4.

used in the literature [30]), it is justified to reject the null hypothesis. In other words, the results are significant at the 5% significance level when $F > F_{\text{critical}}$. The values of p in Table 5 are small enough to discard the null hypothesis except in the case of Table 1. This is as expected, since the DBSCAN variants are designed to obtain a number of clusters as close as possible to that of DBSCAN regardless of the clustering problem at hand.

In Table 1, DBSCAN is the method that obtains the correct number of clusters according to the definitions in Section 2. The six variants of DBSCAN are based on reducing the number of region queries at the expense of splitting some of the DBSCAN clusters. The following conclusions can be drawn from Table 1:

- Borah and Bhattacharyya, the creators of IDBSCAN, remark in [8, page 95] that IDBSCAN finds the same number of clusters as DBSCAN in all their experiments. However, Table 1 shows that this is not true for some of the datasets used in this section. Specifically, IDBSCAN is inferior to DBSCAN in terms of clustering quality for all the instances rBubbles$N.S.L$. Given this interesting result, an immediate question arises: Is there a way to improve IDBSCAN? As shown below, the present work provides an affirmative response to this important question through the introduction of I*DBSCAN.

- In general, I*DBSCAN outperforms the rest of DBSCAN variants regarding the mean number of clusters reported in Table 1. (There is only one dataset instance, rBubbles500096.128.2, for which I*DBSCAN obtains a slightly worse mean number of clusters than that of IDBSCAN.) The superior behavior of I*DBSCAN with respect to the rest of variants is confirmed by Table 4, in which I*DBSCAN obtains remarkable results regarding the mean number of cluster updates.

- I*DBSCAN and IDBSCAN exhibit a slightly inferior behavior to that of DBSCAN. The only dataset instances where I*DBSCAN and IDBSCAN split clusters in comparison to DBSCAN are rBubbles$N.S.L$. Nonetheless, even for these instances, the mean number of clusters for DBSCAN, I*DBSCAN, and IDBSCAN differ only in an amount less than unity in five out of the eight cases. Actually, a contribution of the experimental evaluation performed in the present work with respect to that in [8] is that some instances have been found (those corresponding to rBubbles$N.S.L$) for which IDBSCAN is inferior to DBSCAN.

- Like I*DBSCAN, FDBSCAN#2 shows a good performance. However, FDBSCAN#2 is outperformed by I*DBSCAN in the case of the eight rBubbles$N.S.L$ instances and in the case of one of the arbitrary instances.

- Both DBSCANmax and FDBSCAN#1 are clearly outperformed by DBSCAN, I*DBSCAN, IDBSCAN, and FDBSCAN#2 in most of the cases. Nonetheless, DBSCANmax outperforms FDBSCAN#1 for a greater number of dataset instances in comparison.

- DBRS is clearly the worst method by far.

In summary, I*DBSCAN, IDBSCAN and FDBSCAN#2 are robust methods that exhibit a behavior quite close to that of DBSCAN in terms of clustering quality. They are only challenged by complex dataset instances like rBubbles$N.S.L$, where an important number of pairs of bubbles are separated from each other by a distance slightly inferior to ϵ. This fact facilitates that certain of these pairs of bubbles are incorrectly labeled by the DBSCAN variants as belonging to different clusters.

In general, the results in Table 2 show that DBSCAN has a significantly longer mean running time than its variants, especially when the number of objects in the dataset is high. This is due to the fact that, as shown in Table 3, the variants of DBSCAN perform fewer region queries on average. However, since the mean number of cluster updates tends to increase when the number of region queries is reduced (see Tables 3 and 4), the mean running time can deteriorate for specific DBSCAN variants. For example, in the case of certain dataset instances in Table 2, the mean running time of DBRS is even higher than that of DBSCAN as a consequence of its frequent cluster updates.

The experiments in this section demonstrate that I*DBSCAN, IDBSCAN and FDBSCAN#2 offer important advantages as clustering algorithms for two-dimensional problems. On the one hand, unlike the rest of the DBSCAN variants, their clustering quality is only slightly inferior to that of DBSCAN. (Precisely, one contribution of this paper is to experimentally quantify the difference in clustering quality performance between DBSCAN and these three variants.) On the other hand, they have a much shorter running time than that of DBSCAN, especially for large and dense datasets. Nonetheless, in the case of high-dimensional clustering problems, they have the disadvantage that the number of new seeds selected after each region query grows with the dimension d: linearly for FDBSCAN#2 and exponentially for I*DBSCAN and IDBSCAN (see Section 3.2). This increases the running time of these three algorithms with respect to the two-dimensional case.

Finally, from the results reported in this section, it can be concluded that I*DBSCAN offers the following important advantages:

1. As shown in Table 1, I*DBSCAN is the DBSCAN variant that splits the least number of clusters with respect to the rest of variants. Therefore, I*DBSCAN is the variant producing the most similar clustering quality to that of DBSCAN.

2. I*DBSCAN greatly improves the running times of DBSCAN (see Table 2). Furthermore, the running times of I*DBSCAN are comparable to those of the other two best variants regarding the mean number of clusters, IDBSCAN and FDBSCAN#2.

3. The mean percentage of seeds reported for I*DBSCAN is much lower than that of DBSCAN in an important number of cases (see Table 3).

4. As included in Table 4, the mean number of cluster updates reported for I*DBSCAN is better than those of the rest of variants.

5 Conclusion

Clustering is a task of great importance in various scientific fields. Among the state-of-the-art clustering algorithms, DBSCAN is a well-known method that produces clusters of arbitrary shape by relying on a density-based notion of cluster. DBSCAN performs a neighborhood query for every single object

in the dataset, which usually gives rise to low efficiency. This has motivated the appearance of several variants of DBSCAN that intend to improve its efficiency by acting on the way the region query operations are carried out.

In this paper, we empirically evaluate a number of DBSCAN variants which are based on reducing the number of neighborhood queries during execution. In principle, this technique lowers the computational load at the expense of losing clustering quality with respect to that of DBSCAN. This extensive comparative evaluation is one of the two contributions of this paper. The other is the introduction of a new variant called I*DBSCAN which outperforms the rest of the evaluated variants in terms of the mean number of clusters generated.

The empirical results obtained for a diverse set of two-dimensional clustering problems suggest that three of the tested DBSCAN variants (I*DBSCAN, IDBSCAN, and FDBSCAN#2), characterized by selecting only a few new seeds within each neighborhood region, are only slightly inferior to DBSCAN in terms of clustering quality. Furthermore, it is also remarkable that the efficiency of these three variants is greatly improved with respect to that of DBSCAN.

Regarding I*DBSCAN, the novel DBSCAN variant introduced in this paper, it offers important practical advantages. It is the variant that operates most similarly to DBSCAN in terms of the generated clusters. The running times associated to I*DBSCAN are quite promising in comparison with those of DBSCAN and are comparable to those of other competitive DBSCAN variants. I*DBSCAN is also competitive with regards to the mean percentage of seeds generated and the mean number of cluster updates performed.

Even though the dataset instances employed in Section 4.1 cover a broad range of cases in a systematic way, this aspect can always by improved by incorporating new instances from other domains of interest; certainly, a diverse set of real-world domains could benefit from the research presented in this work. Another aspect that remains to be analyzed in detail is the scalability of I*DBSCAN and the other variants in the context of high-dimensional clustering problems.

The present work opens up the following research directions:

- I*DBSCAN outperforms the other DBSCAN variants and is close to DBSCAN in terms of clustering quality. However, there is still room for improvement regarding the design of new variants whose clustering quality resembles more and more that of DBSCAN, while increasing the clustering process efficiency. I*DBSCAN represents a step forward in that direction.

- Due to the efficiency of I*DBSCAN, it could be interesting to study its usefulness as a clustering algorithm for non-stationary objects in order to learn continuously evolving clusters.

- Since the DBSCAN algorithm is based on local operations, the use of parallel computing in the context of I*DBSCAN and the other DBSCAN variants could be investigated in order to improve their computation times for larger dataset instances.

- I*DBSCAN and the other variants could be extended by considering fuzzy clustering [31], which allows data points to potentially belong to multiple clusters.

References

[1] A. Adeleke, I.-K. Hauwau, and O. Tinuke. Cluster analysis of data points using partitioning and probabilistic model-based algorithms. *International Journal of Applied Information Systems*, 7(7):21–26, 2014.

[2] C. C. Aggarwal and C. K. Reddy. *Data Clustering: Algorithms and Applications*. CRC Press, 2014.

[3] R. Ahmed, E. El-Zaza, and W. Ashour. Multi-density DBSCAN using representatives: MDBSCAN-UR. *Computing and Information Systems*, 15(2):1–7, 2011.

[4] K. S. Al-Sultan. A tabu search approach to the clustering problem. *Pattern Recognition*, 28(9):1443–1451, 1995.

[5] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. OPTICS: Ordering points to identify the clustering structure. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'99)*, pages 49–60. ACM, 1999.

[6] A. M. Bakr, N. M. Ghanem, and M. A. Ismail. Efficient incremental density-based algorithm for clustering large databases. *Alexandria Engineering Journal*, 54(4):1147–1154, 2015.

[7] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'90)*, pages 322–331. ACM, 1990.

[8] B. Borah and D. K. Bhattacharyya. An improved sampling-based DBSCAN for large spatial databases. In *Proceedings of the International Conference on Intelligent Sensing and Information Processing (ICISIP 2004)*, pages 92–96. IEEE, 2004.

[9] R. Caballero, M. Laguna, R. Martí, and J. Molina. Scatter tabu search for multiobjective clustering problems. *Journal of the Operational Research Society*, 62(11):2034–2046, 2011.

[10] R. Campello, D. Moulavi, and J. Sander. Density-based clustering based on hierarchical density estimates. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2013)*, pages 160–172. Springer, 2013.

[11] G. Chen, W. Luo, and T. Zhu. Evolutionary clustering with differential evolution. In *Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC 2014)*, pages 1382–1389. IEEE, 2014.

[12] H. Darong and W. Peng. Grid-based DBSCAN algorithm with referential parameters. *Physics Procedia*, 24:1166–1170, 2012.

[13] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[14] M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, and X. Xu. Incremental clustering for mining in a data warehousing environment. In *Proceedings of the 24th International Conference on Very Large Data Bases (VLDB'98)*, pages 323–333. Morgan Kaufmann, 1998.

[15] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 226–231. The AAAI Press, 1996.

[16] R. A. Fisher. *Statistical Methods for Research Workers*. Oliver and Boyd, 1925.

[17] P. Grabusts and A. Borisov. Using grid-clustering methods in data classification. In *Proceedings of the International Conference on Parallel Computing in Electrical Engineering (PARELEC 2002)*, pages 425–426. IEEE, 2002.

[18] P. Hansen and N. Mladenović. J-MEANS: A new local search heuristic for minimum sum of squares clustering. *Pattern Recognition*, 34(2):405–413, 2001.

[19] X. Hu, L. Liu, N. Qiu, D. Yang, and M. Li. A MapReduce-based improvement algorithm for DBSCAN. *Journal of Algorithms & Computational Technology*, 12(1):53–61, 2018.

[20] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1988.

[21] R. A. Jarvis and E. A. Patrick. Clustering using a similarity measure based on shared near neighbors. *IEEE Transactions on Computers*, C22:1025–1034, 1973.

[22] X. Lai and J.-K. Hao. Iterated variable neighborhood search for the capacitated clustering problem. *Engineering Applications of Artificial Intelligence*, 56:102–120, 2016.

[23] B. Liu. A fast density-based clustering algorithm for large databases. In *Proceedings of the 5th International Conference on Machine Learning and Cybernetics (ICMLC 2006)*, pages 996–1000. IEEE, 2006.

[24] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[25] G. J. McLachlan and K. E. Basford. *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker, Inc., 1988.

[26] A. R. Mekky. Fuzzy neighborhood grid-based DBSCAN using representative points. In *Proceedings of the Third International Conference on Data Mining, Internet Computing, and Big Data (BigData2016)*, pages 63–73. SDIWC, 2016.

[27] H. D. Menéndez, D. F. Barrero, and D. Camacho. A genetic graph-based approach for partitional clustering. *International Journal of Neural Systems*, 24(3):1430008, 2014.

[28] C. A. Murthy and N. Chowdhury. In search of optimal clusters using genetic algorithms. *Pattern Recognition Letters*, 17(8):825–832, 1996.

[29] P. Nerurkar, A. Shirke, M. Chandane, and S. Bhirud. Empirical analysis of data clustering algorithms. *Procedia Computer Science*, 125:770–779, 2018.

[30] R. Nuzzo. Scientific method: statistical errors. *Nature*, 506(7487):150–152, 2014.

[31] W. Pedrycz. *Knowledge-Based Clustering: From Data to Information Granules*. Wiley, 2005.

[32] M. Ranjbar and M. R. Mosavi. Simulated annealing clustering for optimum GPS satellite selection. *International Journal of Computer Science Issues*, 9(3):100–104, 2012.

[33] L. Rokach. A survey of clustering algorithms. In O. Maimon and L. Rokach, editors, *Data Mining and Knowledge Discovery Handbook*, pages 269–298. Springer, 2010. Second edition.

[34] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu. Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications. *Data Mining and Knowledge Discovery*, 2(2):169–194, 1998.

[35] S. Z. Selim and K. S. Al-Sultan. A simulated annealing algorithm for the clustering problem. *Pattern Recognition*, 24(10):1003–1008, 1991.

[36] N. A. Shah and R. Paul. Survey on different grid based clustering algorithms of data mining. *International Journal of Advance Research and Innovative Ideas in Education*, 3(1):1118–1123, 2017.

[37] H. Song and J.-G. Lee. RP-DBSCAN: a superfast parallel DBSCAN algorithm based on random partitioning. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'18)*, pages 1173–1187. ACM, 2018.

[38] M. Stonebraker, J. Frew, K. Gardels, and J. Meredith. The SEQUOIA 2000 storage benchmark. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'93)*, pages 2–11. ACM, 1993.

[39] G. Ulutagay and E. Nasibov. Fuzzy and crisp clustering methods based on the neighborhood concept: A comprehensive review. *Journal of Intelligent & Fuzzy Systems*, 23(6):271–281, 2012.

[40] L. Wang and Z.-O. Wang. CUBN: A clustering algorithm based on density and distance. In *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (ICMLC 2003)*, pages 108–112. IEEE, 2003.

[41] X. Wang and H. J. Hamilton. DBRS: A density-based spatial clustering method with random sampling. Technical Report CS-2003-13, Department of Computer Science, University of Regina, Regina, Canada, 2003.

[42] X. Wang and H. J. Hamilton. DBRS: A density-based spatial clustering method with random sampling. In *Proceedings of the 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-03)*, pages 563–575. Springer, 2003.

[43] U. Wilensky. NetLogo. `http://ccl.northwestern.edu/netlogo/`. Center for Connected Learning and Computer Science, Northwestern University, Evanston, IL, 1999.

[44] C. Xiaoyun, M. Yufang, Z. Yan, and W. Ping. GMDBSCAN: Multi-density DBSCAN cluster based on grid. In *Proceedings of the 2008 IEEE International Conference on e-Business Engineering*, pages 780–783. IEEE, 2008.

[45] X. Xu, J. Jäger, and H.-P. Kriegel. A fast parallel clustering algorithm for large spatial databases. *Data Mining and Knowledge Discovery*, 3(3):263–290, 1999.

[46] C. T. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers*, C-20(1):68–86, 1971.

[47] M. Zait and H. Messatfa. A comparative study of clustering methods. *Future Generation Computer Systems*, 13(2-3):149–159, 1997.

[48] T.-T. Zhang and B. Yuan. Density-based multiscale analysis for clustering in strong noise settings with varying densities. *IEEE Access*, 6:25861–25873, 2018.

[49] A.-Y. Zhou, S.-G. Zhou, J. Cao, Y. Fan, and Y. Hu. Approaches for scaling DBSCAN algorithm to large spatial databases. *Journal of Computer Science and Technology*, 15(6):509–526, 2000.

[50] S.-G. Zhou, A.-Y. Zhou, W. Jin, Y. Fan, and W.-N. Qian. FDBSCAN: A fast DBSCAN algorithm. *Journal of Software*, 11(6):735–744, 2000.