DBSCAN parallelization by spatial dataset division and hyperparameter adjustment

Vadim Romanuke[1,*,†], Sergii Pavlov[2,†], Svitlana Yaremko[1,†], Artem Guralnyk[1,†] and Olena Smetaniuk[2,†]

[1] *Vinnytsia Institute of Trade and Economics of State University of Trade and Economics, Soborna str., 87, 21050, Vinnytsia, Ukraine*

[2] *Vinnytsia National Technical University, Khmelnytske sh., 95, 21021, Vinnytsia, Ukraine*

Abstract

An approach to parallelizing the DBSCAN algorithm without losing in accuracy is presented to further improve quality of arbitrary-shape clustering. The dataset is divided into a number of subsets, one of which should be labeled to adjust the DBSCAN hyperparameters — the neighborhood radius and minimum-neighbors number. The neighborhood radius, set to a minimum, is adjusted first, where the original DBSCAN is applied to a subset of points with known ground truth labels. While the current accuracy of clustering is not improved, the neighborhood radius is increased by an increment step. The original DBSCAN algorithm is applied with the best neighborhood radius to the remaining subsets. The minimum-neighbors number, set to 3, is adjusted second by the best neighborhood radius. Compared to the performance of the original DBSCAN algorithm, the parallelized DBSCAN performs more accurately being extremely fast. As the dataset size increases and the number of clusters increases, one by one or together, the parallelized DBSCAN outperforms the original DBSCAN more. However, the DBSCAN parallelization by spatial dataset division and hyperparameter adjustment is less efficient on datasets with more complex structure. Nevertheless, the average speedup exceeds 80 times for planar datasets and exceeds 300 times for three-dimensional datasets.

Keywords

arbitrary-shape clustering, DBSCAN, parallelization, hyperparameter adjustment, neighborhood radius, spatial dataset division

1. Introduction

DBSCAN is one of the most commonly used clustering algorithms whose purpose is to reveal clusters of arbitrary shape by domain minimal knowledge [1, 2]. DBSCAN groups

BAIT'2024: The 1st International Workshop on "Bioinformatics and applied information technologies", October 02-04, 2024, Zboriv, Ukraine

* Corresponding author.

† These authors contributed equally.

romanukevadimv@gmail.com (V. V. Romanuke); psv@vntu.edu.ua (S. V. Pavlov); svitlana_yaremko@ukr.net (S. A. Yaremko); artem.guralnyk@gmail.com (A. B. Guralnyk); smetaniuk@vntu.edu.ua (O. A. Smetaniuk)

0000-0001-9638-9572 (V. V. Romanuke); 0000-0002-0051-5560 (S. V. Pavlov); 0000-0002-0605-9324 (S. A. Yaremko); 0000-0003-1977-6338 (A. B. Guralnyk); 0000-0001-5207-6451 (O. A. Smetaniuk)

© 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR-WS.org/Vol-3842/paper8.pdf

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

points based on the notion of ε-neighborhood of a point and a specified minimum number of neighboring points $q_{\min}$, which further are supplemented with the notions of core points, directly reachable points, non-directly reachable points, and outliers [3, 4]. Besides, to rectify the non-symmetric relation of the reachability [5, 6], a symmetric notion of density-connectedness is added [7, 8]. The density is defined by the neighborhood radius ε and number $q_{\min}$, where a cluster contains at least $q_{\min}$ points. The ε-neighborhood is defined as a set of points falling within radius ε by a distance function [5, 9, 10]. Apart from DBSCAN hyperparameters ε and $q_{\min}$, the distance function is seen as an additional parameter of the DBSCAN algorithm [7, 11, 12].

For N points in a dataset, the DBSCAN average runtime complexity is $O(N \log N)$, although the worst case of $O(N^2)$ is not excluded [3, 4, 7], especially when there is probable incompleteness in data [9, 10, 13] or the neighborhood radius can vary in a wide range [5, 12, 14]. Whereas the overall performance of DBSCAN is quite satisfactory, and it successfully reveals outliers and identifies nonglobular shapes (contrary to other commonly used clustering algorithms like, e. g., *k*-means [9, 15, 16]), it fails to handle datasets whose clusters are of much varying densities [17, 18]. The second drawback is the neighborhood radius selection [8, 11, 14]. While datasets of planar points or points in the three-dimensional space are visualized, the neighborhood radius is well understood and assessed, but higher dimensions are problematic to "see" a meaningful distance [19, 20]. Moreover, quality of features may be pretty different and may change through a domain which is clustered [8]. This hinders in obtaining accurately partitioned datasets by coarsely applying DBSCAN [17]. There are domains whose factual clusters have multiple meaningful radii of the neighborhood [11, 13, 20].

Both the weaknesses of DBSCAN are addressed by OPTICS [21, 22], which is another algorithm whose purpose is to detect meaningful clusters in data of varying density [23, 24]. OPTICS also requires two hyperparameters ε and $q_{\min}$, although the neighborhood radius is not necessary. The radius can be set to the maximum possible value, but this results in runtime close to $O(N^2)$ [25]. Besides, OPTICS is mainly intended for hierarchical clustering, and OPTICS is reported to have an actual constant slowdown factor of 1.6 compared to DBSCAN [21, 26].

Another DBSCAN drawback, being reported less frequently, is its poorer performance on large datasets [27]. This especially concerns high-dimensional data, where it is difficult to find an appropriate value for ε due to adjusting this hyperparameter takes significant amounts of time and computational resources. Overall, tunability of DBSCAN hyperparameters (as well in other clustering algorithms) worsens as the dataset enlarges [28, 29].

Nevertheless, when a large dataset is assumed to have tightly connected or not sparsely scattered clusters, it is possible to divide the dataset into multiple subdatasets in order to apply the DBSCAN algorithm to every subdataset separately. This resembles divide-and-conquer approach successfully applied to solving large combinatorial optimization problems like the traveling salesman problem with thousands of nodes and larger [30]. In particular, the traveling salesman problem nodes are grouped into a definite number of

clusters, each of which corresponds to an open-tour traveling salesman subproblem subsequently solved independently of solving other subproblems [31, 32].

2. Problem statement

In cluster analysis, spatial dataset division is effective when possible clusters are not sparsely scattered. Otherwise, the problem of clustering tends to be almost trivial — the distinctly distanced groups of objects can be more efficiently separated by the support vector machine (linearly inseparable data are projected onto a higher dimensional space to become linearly separable [33, 34]) or other similar approaches including *k*-means with evaluating the optimal number of clusters using the silhouette clustering evaluation criterion [15, 35, 36]. To the contrary, potential clusters closely packed together are totally indistinguishable without considering spatial densities if, for instance, the clusters have at least a partially hierarchical structure [2, 37, 38]. A marginal example of such a structure is a dataset consisting of non-overlapping enclosed hyperspheres or hyperellipsoids (for planar objects, they are non-overlapping enclosed circles or ellipses), or objects of similar form being irregularly distorted or nonconvex [29, 39]. Another major example frequently occurred in practice is a dataset consisting of cluster bands, where serpentine-like clusters have a resembling structure [40].

Naturally, the DBSCAN algorithm could have been easily sped up if it was generally parallelizable [1, 2, 7]. Unfortunately, the algorithm is of two loops, and there is the nested loop which cannot be made independent of the outside loop [1, 4]. The outside loop, working with each point in a given dataset, is based on the range query which can be implemented using database indexing for better performance rather than a slow linear scanning. Although some efforts have been made in this direction to parallelize the outside loop, the gain does not seem to have a significant impact [2, 17, 19].

A follow-up DBSCAN algorithm named HDBSCAN [2, 17] was an attempt to rectify the DBSCAN flat-result drawbacks rather its poor parallelizability. Neither did GDBSCAN advances in parallelizability [5, 7], which only moved the original DBSCAN algorithm parameters ε and $q_{\min}$ to the predicates.

Issuing from unsatisfactory runtime complexity of the DBSCAN algorithm for large datasets, the goal is to suggest an approach to DBSCAN parallelization without losing much in accuracy. The approach should be algorithmized by automatically adjusting neighborhood radius ε to the most efficient value. A series of computational experiments is to be carried out in order to see the performance of the parallelized DBSCAN algorithm with the hyperparameter adjustment compared to the performance of the original DBSCAN algorithm. The comparative results are then to be discussed by underlining limitations and drawbacks of the suggested DBSCAN parallelization.

3. DBSCAN parallelization

To adjust DBSCAN hyperparameters automatically, consider an ultimate repetition of trying to improve the accuracy. As there are more possible versions of the neighborhood radius than those of the minimum number of neighboring points, the neighborhood radius

is adjusted first. For this, denote the maximum number of accuracy improvement fails by $F_{\max}(\varepsilon)$. While the minimum number of neighboring points is adjusted, the maximum number of accuracy improvement fails denoted by $F_{\max}(q_{\min})$ is used. These hyperparameters are adjusted as follows:

1. Divide the dataset into S subsets.

2. Set ε to a minimum and apply the DBSCAN algorithm to a subset of points with known ground truth labels.

3. While the current number of accuracy improvement fails is fewer than $F_{\max}(\varepsilon)$, increase ε by $\Delta\varepsilon$. Once the number of outliers and redundant labels becomes fewer, set the current number of accuracy improvement fails to 0, and store the current value of ε as the best one; otherwise, increase the current number of accuracy improvement fails by 1.

4. Denote the best neighborhood radius by ε^*.

5. Apply the DBSCAN algorithm to the remaining $S-1$ subsets.

6. Set $q_{\min}$ to 3 and apply the DBSCAN algorithm to the subset by the best neighborhood radius ε^*.

7. While the current number of accuracy improvement fails is fewer than $F_{\max}(q_{\min})$, increase $q_{\min}$ by 1. Once the number of outliers and redundant labels becomes fewer, set the current number of accuracy improvement fails to 0, and store the current value of $q_{\min}$ as the best one; otherwise, increase the current number of accuracy improvement fails by 1.

Obviously, running the DBSCAN algorithm on the remaining $S-1$ subsets can be executed in parallel. Moreover, as the hyperparameters are adjusted (on a labeled subset), the size of a labeled subset can be set to a few different versions, and the adjustment can be executed in parallel on these differently sized subsets [16, 37, 38]. Differently sized subsets are obtained by changing number S.

First of all, the DBSCAN parallelization is expected to significantly reduce the computation time. Secondly, the clustering accuracy is believed to remain comparable to the DBSCAN algorithm itself, if even the adjusted hyperparameters are used in it. The correctness of these hypotheses is going to be ascertained or falsified by computational experiments. A specificity of such experiments is that the dataset can be visualized, with better or worse extent of cluster appearance and distinguishability.

4. Computational experiments

The performance of the parallelized DBSCAN algorithm with the hyperparameter adjustment is estimated by running the above-stated algorithm on planar datasets whose potential clusters are closely packed together. There are two types of such datasets: developing and enveloping. Serpentine-like clusters form a developing dataset (Figure 1). Non-overlapping enclosed circles or ellipses modulated by sinusoidal functions form an enveloping dataset (Figure 2). For abbreviation, these clusters (datasets) will be called serpentine and circular clusters (datasets), respectively.

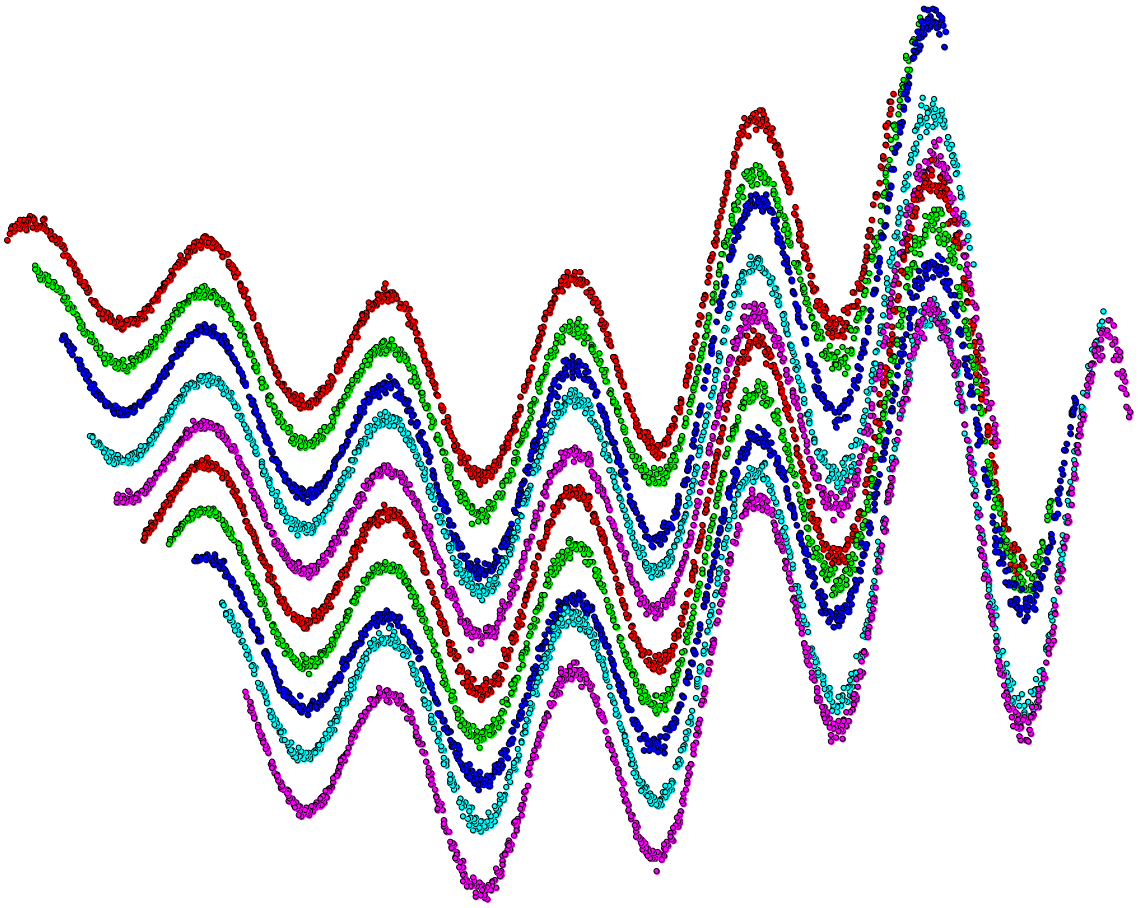

Figure 1: An example of a dataset comprised by 10 serpentine-like clusters (each cluster contains 1000 points).

A serpentine dataset point $[x_i \quad y_i] \in \mathbb{R}^2$ is randomly generated using values

$$\xi_{ml},\ \theta_{ml},\ \vartheta_{ml},\ \zeta_{ml}$$

of normally distributed random variables with zero mean and unit variance for point l belonging to cluster m, where

$$m = \overline{1, M} \text{ and } l = \overline{1, L},\ i = l + (m-1)\cdot L, \tag{1}$$

i. e., $i = \overline{1, N}$, the dataset size is $N = M \cdot L$, and M is the number of clusters of size L each. Besides, values η and μ of two additional normally distributed random variables with zero mean and unit variance are used for this dataset. Thus, the horizontal component is

$$x_i = \frac{100\pi}{L-1}\cdot(l-1) + a_x\xi_{ml} + 3m\pi \tag{2}$$

by (1), where a_x is chosen randomly between 0.1 and 1 with a step of 1110^{-1}. The vertical

component is

$$y_i = \left(a_{1ml}\sin\left(\omega_1 x_i\right) + a_{2ml}\cos\left(\omega_2 x_i\right)\right)\cdot e^{bx_i} + \\ + c_1\zeta_{ml} + c_2 x_i - 4\cdot\left(m-1\right) \tag{3}$$

by (2), (1),

$$a_{1ml} = a_{01ml} + 0.125\theta_{ml}, \tag{4}$$

$$a_{2ml} = a_{02ml} + 0.125\vartheta_{ml}, \tag{5}$$

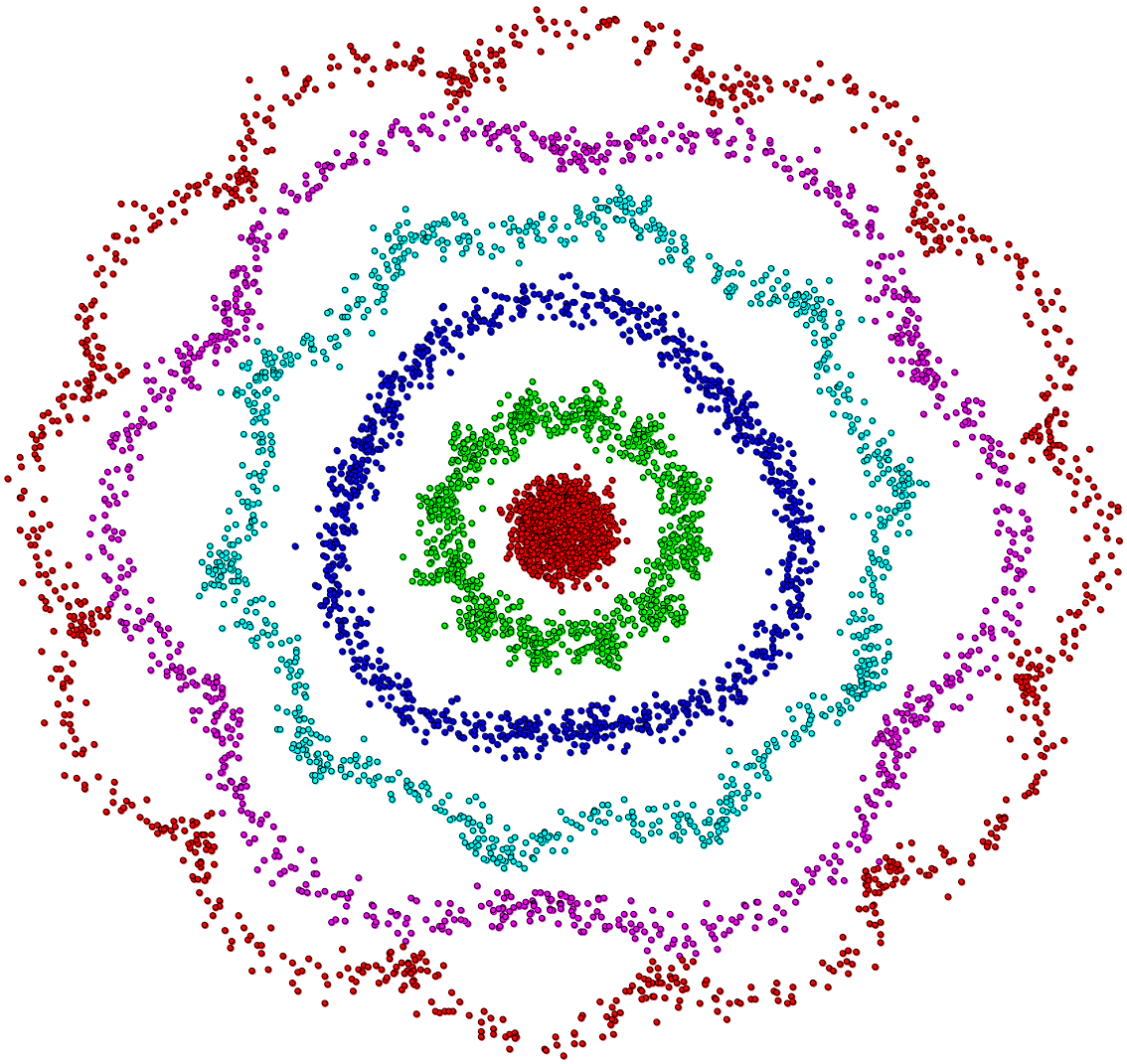

Figure 2: An example of a dataset comprised by six enclosed circular clusters (each cluster contains 1000 points).

where both a_{01ml} and a_{02ml} are independent being chosen randomly between 3 and 4 with

a step of 0.001; both ω_1 and ω_2 are independent being chosen randomly between 0.01 and 0.1 with a step of 11100^{-1}; b is chosen randomly between 0.0005 and 0.005 with a step of 222000^{-1}, whereupon it is set negative if $\eta<0$; c_1 is varied between 0.1 and 0.5 with a step of 0.1; c_2 is chosen randomly between 0.0005 and 0.005 with a step of 222000^{-1}, whereupon it is set negative if $\mu<0$.

The serpentine dataset by (2), (3), (1), (4), (5) is generated with three to 18 clusters, $M=\overline{3,18}$, for five versions of noise magnitude,

$$c_1 \in \{0.1n\}_{n=1}^{5},$$

where each cluster contains 1000 to 10000 points with a step of 1000 points,

$$L \in \{1000k\}_{k=1}^{10}.$$

Clusters contain the same number of points $L=N/M$. The dataset is divided into S subsets so that each cluster would contain the same number of points within every subset, where

$$S \in \{L/10, L/20, L/40, L/50, L/100\} \tag{6}$$

and subset s_1 is clustered, $s_1=S/2$ if S is even and

$$s_1=(S+1)/2$$

if S is odd. Therefore, for a one generation of the serpentine dataset, there are 10 versions of the dataset size, 16 versions of the dataset cluster number, five versions of noise magnitude, and five versions of the dataset division. Applied to a dataset, the parallelized DBSCAN algorithm returns the best neighborhood radius ε^*, the best minimum number of neighboring points $q_{\min}^*$, the number of outliers N_{outliers} (labeled by -1), the number of points incorrectly labeled as non-existing clusters N_{other} (their labels are greater than M, as if there are some other one or a few clusters), and the number of points incorrectly labeled as existing clusters N_{confused} (being between 1 and M, their labels are factually incorrect or confused). The total number of missed (incorrectly labeled) points is

$$N_{\text{missed}}=N_{\text{outliers}}+N_{\text{other}}+N_{\text{confused}} \tag{7}$$

whose percentage is

$$\rho=100N_{\text{missed}}/N. \tag{8}$$

The serpentine dataset generation is repeated for 100 times, whereupon the results are averaged over the 100 repetitions. The averages are presented as a $10\times16\times5\times5$ array

whose entry is a set of

$$\varepsilon^{*},\ q_{\min}^{*},\ N_{\text{outliers}},\ N_{\text{other}},\ N_{\text{confused}},\ N_{\text{missed}},\ \rho. \tag{9}$$

Another $10\times16\times5\times5$ array presents averages of the computation time. To compare results (9) with those produced by the DBSCAN algorithm itself, the parallelized DBSCAN algorithm with the hyperparameter adjustment is also used, but with $S=1$ and $s_1=1$. The original DBSCAN algorithm results are stored as a $10\times16\times5$ array with (9). Another $10\times16\times5$ array presents averages of the DBSCAN computation time.

Comparative computation time statistics for the planar serpentine dataset is presented in Table 1, where every entry is an averaged ratio of the original DBSCAN algorithm computation time to the parallelized DBSCAN algorithm computation time. Ratio $\tau_{\text{noise}}(N,M)$ is a computation time function of the dataset size and the number of dataset clusters, which is a 10×16 matrix preliminarily averaged over noise; $\tau_{\text{size}}(M,c_1)$ is a 16×5 matrix preliminarily averaged over dataset size; $\tau_{M}(N,c_1)$ is a 10×5 matrix preliminarily averaged over the number of dataset clusters. The advantage of the parallelized DBSCAN algorithm is quite obvious — it is much faster, while, prudently speaking, the speedup minimum is above 10 times and the speedup maximum exceeds 100 times. On overall average, the parallelized DBSCAN is 97.21 times faster.

Table 1
Comparative computation time statistics for the planar serpentine dataset

Statistics	Minimum	Average	Maximum	Standard deviation
$\tau_{\text{noise}}(N,M)$	13.78549	82.3331	157.3137	35.78136
$\tau_{\text{size}}(M,c_1)$	52.49892	91.56534	117.2529	16.80617
$\tau_{M}(N,c_1)$	22.08231	87.4891	139.2775	34.42707

Comparative performance statistics for the planar serpentine dataset is presented in Table 2, where rows with

$$N_{\text{outliers}},\ N_{\text{other}},\ N_{\text{confused}} \tag{10}$$

contain percentages, “Five intervals” stands for the parallelized DBSCAN with five versions of the dataset division, and “Best interval” stands for the parallelized DBSCAN with the version at which percentage (8) of missed (incorrectly labeled) points is minimal. Herein and below, the DBSCAN accuracy is decomposed into rates for (10) and (8). The parallelized DBSCAN misses (loses) points at a rate of about 7 %, whereas it is above 26 % by the original DBSCAN. Points misidentified as outliers are lost at almost equal rates, although a little advantage of the parallelized DBSCAN exists whose rate is below 1.95 % (while the original DBSCAN loses above 2.1 % of points misidentified as outliers). The

original DBSCAN incorrectly assigns above 3.1 % of points to non-existing clusters, but this rate is as 2.5 times as lower for the parallelized DBSCAN. The original DBSCAN confuses clusters even more exceeding 21 % of points incorrectly assigned to existing clusters, while this rate for the parallelized DBSCAN is slightly above 3.8 %. When the best interval is selected, the mentioned rates for (10) and (8) drop below 0.21 %, 0.26 %, 0.73 %, 1.19 %, respectively. However, the worst cases of the rates for (10) and (8) are not excluded even by the parallelized DBSCAN with the best interval (Figure 3).

Table 2
Comparative performance statistics for the planar serpentine dataset

Statistics		Minimum	Average	Maximum	Standard deviation
ε^*	DBSCAN	0.04	0.93991	2.87	0.3686
	Five intervals	0.06	1.1708	3.53	0.43709
	Best interval	0.31	1.34179	3.53	0.46328
$q^*_{\min}$	DBSCAN	3	4.03313	27	2.39181
	Five intervals	3	3.0225	17	0.34669
	Best interval	3	3.00113	9	0.06324
N_{outliers}	DBSCAN	0	2.16369	99.90769	13.575
	Five intervals	0	1.93919	99.55	10.98534
	Best interval	0	0.20774	66.50909	0.8341
N_{other}	DBSCAN	0	3.13743	82.32462	8.84055
	Five intervals	0	1.22586	79.81176	3.53786
	Best interval	0	0.25967	31.83636	1.03566
N_{confused}	DBSCAN	0	21.45342	94.44389	28.24494
	Five intervals	0	3.80889	80.86471	8.94931
	Best interval	0	0.72118	77.975	2.92216
ρ	DBSCAN	0	26.75454	99.97692	33.20733
	Five intervals	0	6.97394	99.9	16.30767
	Best interval	0	1.1886	87.35455	4.06603

A circular dataset point $[x_i \quad y_i] \in \mathbb{R}^2$ is randomly generated using values ξ_{ml}, θ_{ml} of normally distributed random variables with zero mean and unit variance and values ζ_{ml}, ϑ_{ml} of uniformly distributed random variables on interval $(0;1)$ for point l belonging to cluster m by (1), where

$$x_i = a_m \sin\left(\frac{2\pi}{L-1}\cdot(l-1)\right) + d_m b_m \cos\left(\omega_m \cdot \frac{2\pi}{L-1}\cdot(l-1) - \varphi_m\right) + c_1(\xi_{ml} + \zeta_{ml}) \tag{11}$$

and

$$y_i = a_m \cos\left(\frac{2\pi}{L-1}\cdot(l-1)\right) + h_m b_m \sin\left(\omega_m \cdot \frac{2\pi}{L-1}\cdot(l-1) - \varphi_m\right) + c_1(\theta_{ml} + \vartheta_{ml}) \tag{12}$$

by

$$\{a_m\}_{m=1}^{8} = \{1, 11, 20.5, 29.5, 38, 46, 53.5, 60.5\}, \tag{13}$$

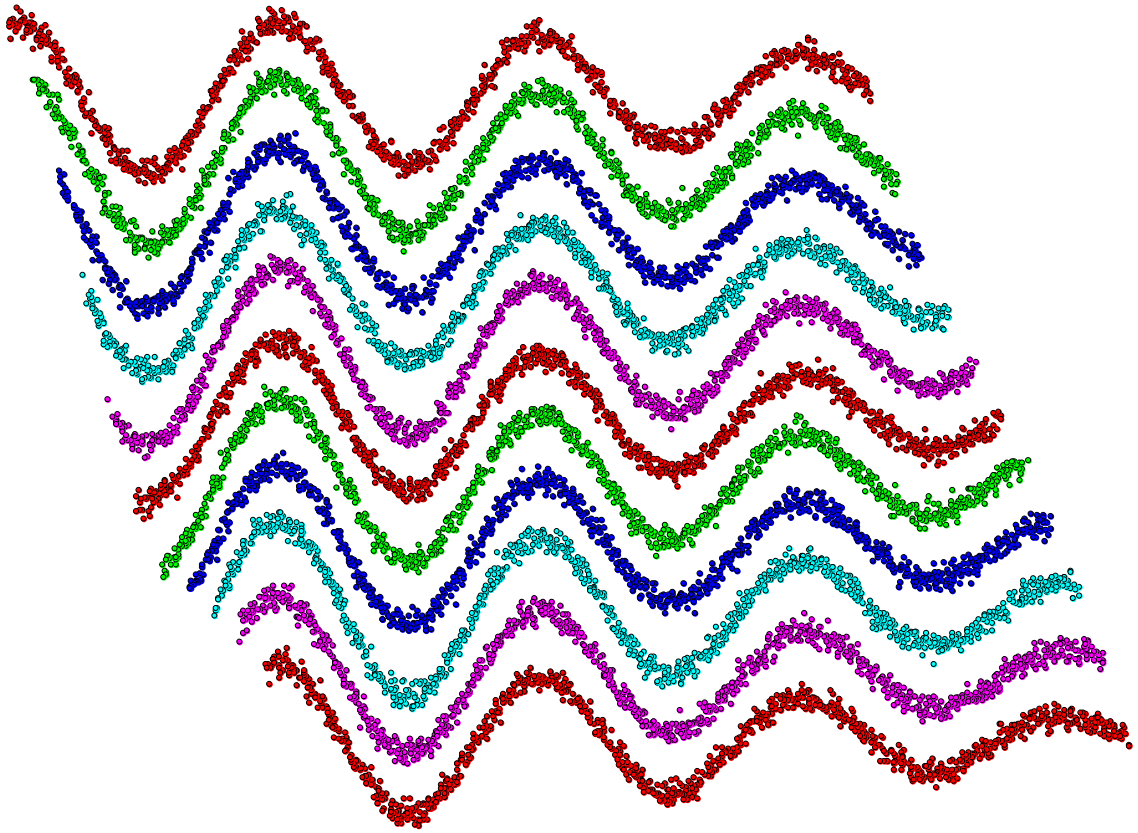

Figure 3: The worst case of the planar serpentine dataset of 11000 points and 11 clusters, where the best result of $\rho = 87.35455$ is produced by the best-interval parallelized DBSCAN.

and values η and μ of two additional normally distributed random variables with zero mean and unit variance, wherein b_m is chosen randomly between 0.1 and 3 with a step of 0.1; $d_m = 1$ if $\eta > 0$ and $d_m = -1$ if $\eta < 0$; ω_m is chosen randomly between 2 and 18 with a step of 1; φ_m is chosen randomly between 0 and π with a step of $\frac{\pi}{999}$; $h_m = 1$ if $\mu > 0$ and $h_m = -1$ if $\mu < 0$; c_1 is varied between 0.2 and 1 with a step of 0.2.

The planar circular dataset by (11), (12), (1), (13) is generated with three to eight clusters, $M = \overline{3,8}$, for five versions of noise magnitude,

$$c_1 \in \{0.2n\}_{n=1}^{5},$$

where each cluster contains 1000 to 10000 points with a step of 1000 points,

$$L \in \{1000k\}_{k=1}^{10}.$$

The remaining parameters of the circular dataset and its processing including (6) — (9) are the same as for the serpentine dataset, except for the array size that is $10 \times 6 \times 5 \times 5$ now (the second dimension has 6 entries, which influences the subsequent arrays upon averaging, minimizing, and maximizing).

Presented in Table 3, comparative computation time statistics for the planar circular dataset resemble that in Table 1 for the planar serpentine dataset. Thus, the parallelized DBSCAN speedup minimum is, prudently speaking, above 15 times and the speedup maximum exceeds 125 times, by roughly the same standard deviations. On overall average, the parallelized DBSCAN is 99.67 times faster (particularly, it is due to the maximum number of clusters here is fewer than that for the planar serpentine dataset).

Table 3
Comparative computation time statistics for the planar circular dataset

Statistics	Minimum	Average	Maximum	Standard deviation
$\tau_{\text{noise}}(N, M)$	17.38579	87.96091	151.3004	38.82666
$\tau_{\text{size}}(M, c_1)$	71.587	99.31722	137.7623	16.79144
$\tau_M(N, c_1)$	18.54298	88.55638	169.5846	40.77947

Presented in Table 4, comparative performance statistics for the planar circular dataset confirms the advantage of the parallelized DBSCAN algorithm. The parallelized DBSCAN loses points at a rate of about 7.12 % (slightly higher than that for the planar dataset), whereas it is 11.85 % by the original DBSCAN being 2.25 times lower than that for the planar dataset. Points misidentified as outliers are lost at rates of 1 % and 1.42 % by the original and parallelized DBSCAN algorithms, respectively. So, the parallelized DBSCAN may lose more circularly located points than the original DBSCAN. The original DBSCAN

incorrectly assigns above 5.59 % of points to non-existing clusters, but this rate is as 4.26 times as lower for the parallelized DBSCAN. The cluster confusion is at similar rates — it is 5.25 % and 4.39 % by the original and parallelized DBSCAN algorithms, respectively. When the best interval is selected, the mentioned rates for (10) and (8) drop below 0.7 %, 0.3 %, 1.97 %, 2.96 %, respectively. These rates are clearly higher than those for the planar serpentine dataset. Nevertheless, the planar circular dataset worst case (Figure 4) is not a complete fail like the planar serpentine dataset worst case is.

Table 4
Comparative performance statistics for the planar circular dataset

Statistics		Minimum	Average	Maximum	Standard deviation
ε^*	DBSCAN	0.34	0.99729	2.46	0.32409
	Five intervals	0.01	1.20794	3.51	0.53009
	Best interval	0.31	1.33544	3.37	0.55512
$q^*_{\min}$	DBSCAN	3	4.68783	30	3.56671
	Five intervals	3	3.10963	28	0.95433
	Best interval	3	3.16067	20	1.22944
N_{outliers}	DBSCAN	0	1.01034	35.72857	2.6169
	Five intervals	0	1.41552	100	4.8618
	Best interval	0	0.69079	20.01563	1.21654
N_{other}	DBSCAN	0	5.59503	67.76286	12.54719
	Five intervals	0	1.3124	46.65625	3.22294
	Best interval	0	0.29938	11.75	0.8674
N_{confused}	DBSCAN	0	5.24657	74.58056	10.69118
	Five intervals	0	4.38885	67.45	7.23762
	Best interval	0	1.96513	29.9125	3.54033
ρ	DBSCAN	0	11.85194	80.95	19.0927
	Five intervals	0	7.11677	100	10.6957
	Best interval	0	2.9553	44.9	4.55293

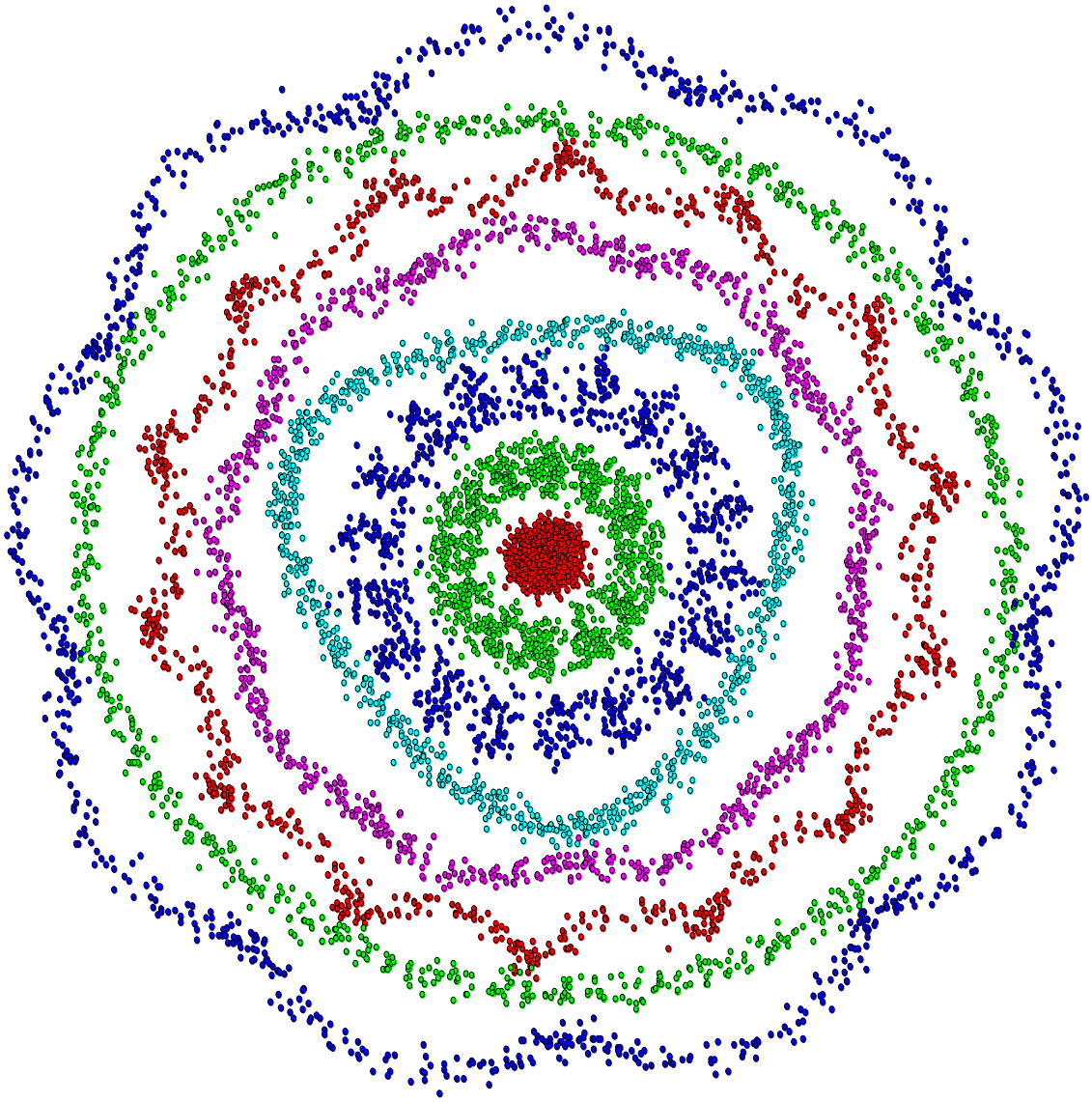

Figure 4: The worst case of the planar circular dataset of 8000 points and 8 clusters, where the best result of $\rho = 44.9$ is produced by the best-interval parallelized DBSCAN.

It is worth noting that the dataset worst case in Figure 4 has far much more variety in its structure than that in the worst case in Figure 3. Despite this, the best-interval parallelized DBSCAN performs relatively much better than the original DBSCAN. The latter has 73.7 % of missed points versus 44.9 % of points missed by the parallelized DBSCAN with $S = 100$ (Figure 5). The percentage of outliers, however, is 13.1 % by the parallelized DBSCAN, whereas the original DBSCAN leaves here 7.225 % of points as outliers. Meanwhile, the original DBSCAN assigns 59.925 % of points to non-existing clusters, and this rate is just 8.3 % for the parallelized DBSCAN (the black-square marked points are clearly seen in Figure 5). Due to this, the cluster confusion is lower at the original DBSCAN — it is 6.55 % versus 23.5 % at the parallelized DBSCAN.

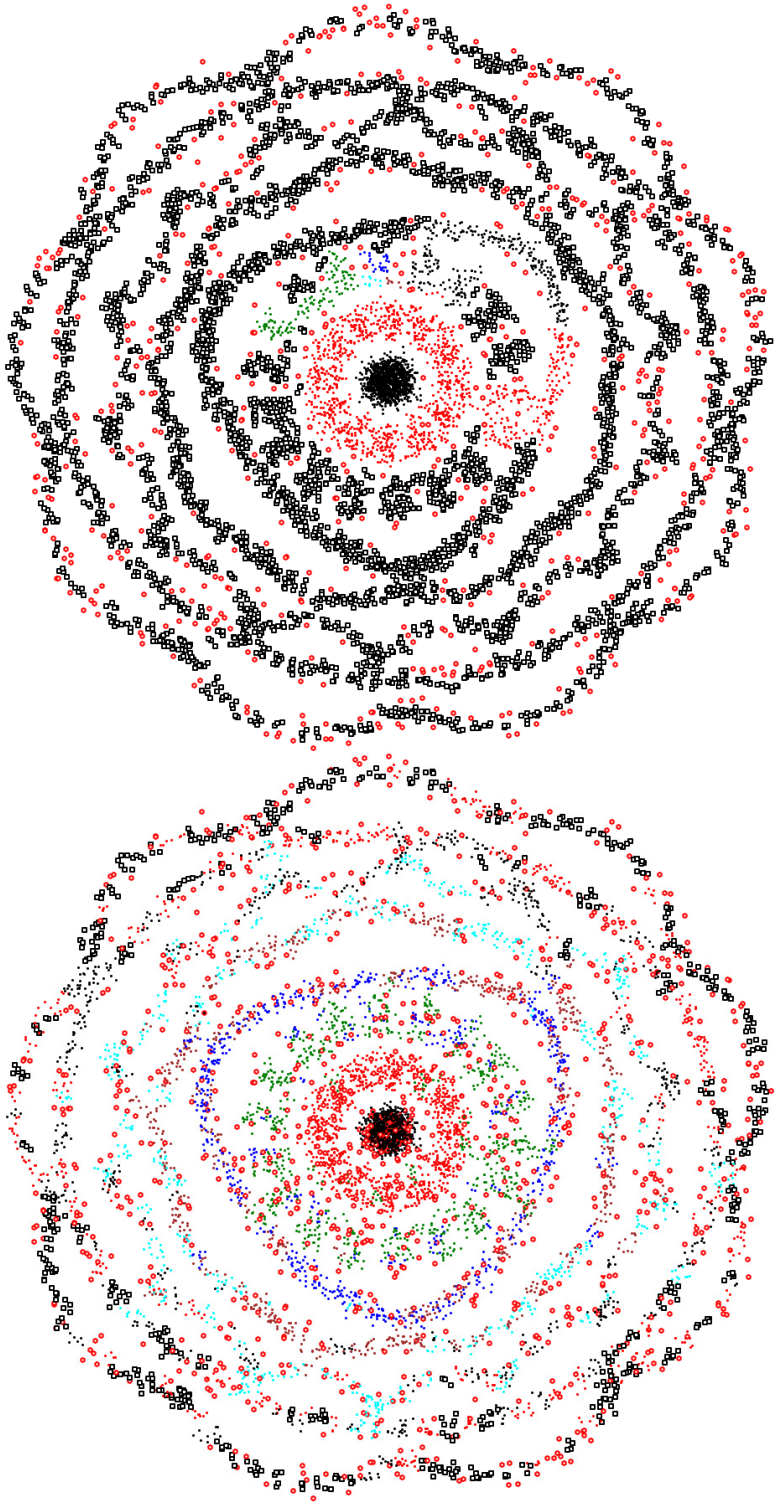

Figure 5: The original DBSCAN performance (above) versus the best-interval parallelized DBSCAN performance (below) in the worst case of the planar circular dataset in Figure 4 (black squares mark points assigned to non-existing clusters, and red squares mark outliers).

In this particular example, the parallelized DBSCAN performance significantly worsens as the dataset is divided into fewer subsets. Thus, $\rho = 48.075$ and the rates for (10) are 6.7625 %, 13.1625 %, 28.15 % with $S = 50$, but $\rho = 63.6625$ and the rates for (10) are 5.9875 %, 26.9125 %, 30.7625 % with $S = 25$ (Figure 6). Furthermore, these rates are 52.9625 %, 1.1875 %, 3.2625 %, 48.5125 % with $S = 20$, and 69.25 %, 0.7875 %, 1.0125 %, 67.45 % with $S = 10$, respectively (Figure 7), i. e. the rate of the cluster confusion grows while the rate of outliers drops. The rate of points assigned to non-existing clusters becomes lower as the subset is made larger (or, in other words, number S is taken fewer).

For creating three-dimensional serpentine-like clusters, a circular dataset point

$$\begin{bmatrix} x_i & y_i & z_i \end{bmatrix} \in \mathbb{R}^3$$

is randomly generated using values

$$\mu_{ml},\ \lambda_{ml},\ \xi_{ml},\ \theta_{ml},\ \nu_{ml}$$

of normally distributed random variables with zero mean and unit variance and values

$$\zeta_{ml},\ \vartheta_{ml},\ \upsilon_{ml}$$

of uniformly distributed random variables on interval $(0;1)$ for point l belonging to cluster m by (1), where

$$x_i = \left(a_m \sin\left(\overline{\alpha}_{ml}\right) + d_m b_m \cos\left(\omega_m \overline{\alpha}_{ml} - \varphi_m\right)\right)\sin\left(\overline{\beta}_{ml}\right) + c_1\left(\xi_{ml} + \zeta_{ml}\right), \tag{14}$$

$$y_i = \left(a_m \cos\left(\overline{\alpha}_{ml}\right) + h_m b_m \sin\left(\omega_m \overline{\alpha}_{ml} - \varphi_m\right)\right)\sin\left(\overline{\beta}_{ml}\right) + c_1\left(\theta_{ml} + \vartheta_{ml}\right), \tag{15}$$

$$z_i = \left(a_m + g_m b_m \sin\left(\omega_m \overline{\alpha}_{ml} - \varphi_m\right)\right)\cos\left(\overline{\beta}_{ml}\right) + c_1\left(\nu_{ml} + \upsilon_{ml}\right) \tag{16}$$

by (13),

$$\overline{\alpha}_{ml} = \frac{8\pi}{L-1}\cdot(l-1) + 0.1\alpha_{ml},\ \overline{\alpha}_{ml} < \overline{\alpha}_{m,l+1}\ \ \forall l = \overline{1,L}, \tag{17}$$

$$\overline{\beta}_{ml} = \frac{5\pi}{L-1}\cdot(l-1) + 0.1\beta_{ml},\ \overline{\beta}_{ml} < \overline{\beta}_{m,l+1}\ \ \forall l = \overline{1,L}, \tag{18}$$

and values η, μ, λ of three additional normally distributed random variables with zero mean and unit variance, wherein b_m is chosen randomly between 0.1 and 3 with a step of 0.1; $d_m = 1$ if $\eta > 0$ and $d_m = -1$ if $\eta < 0$; ω_m is chosen randomly between 2 and 18 with a step of 1; φ_m is chosen randomly between 0 and π with a step of $\frac{\pi}{999}$; $h_m = 1$ if $\mu > 0$ and

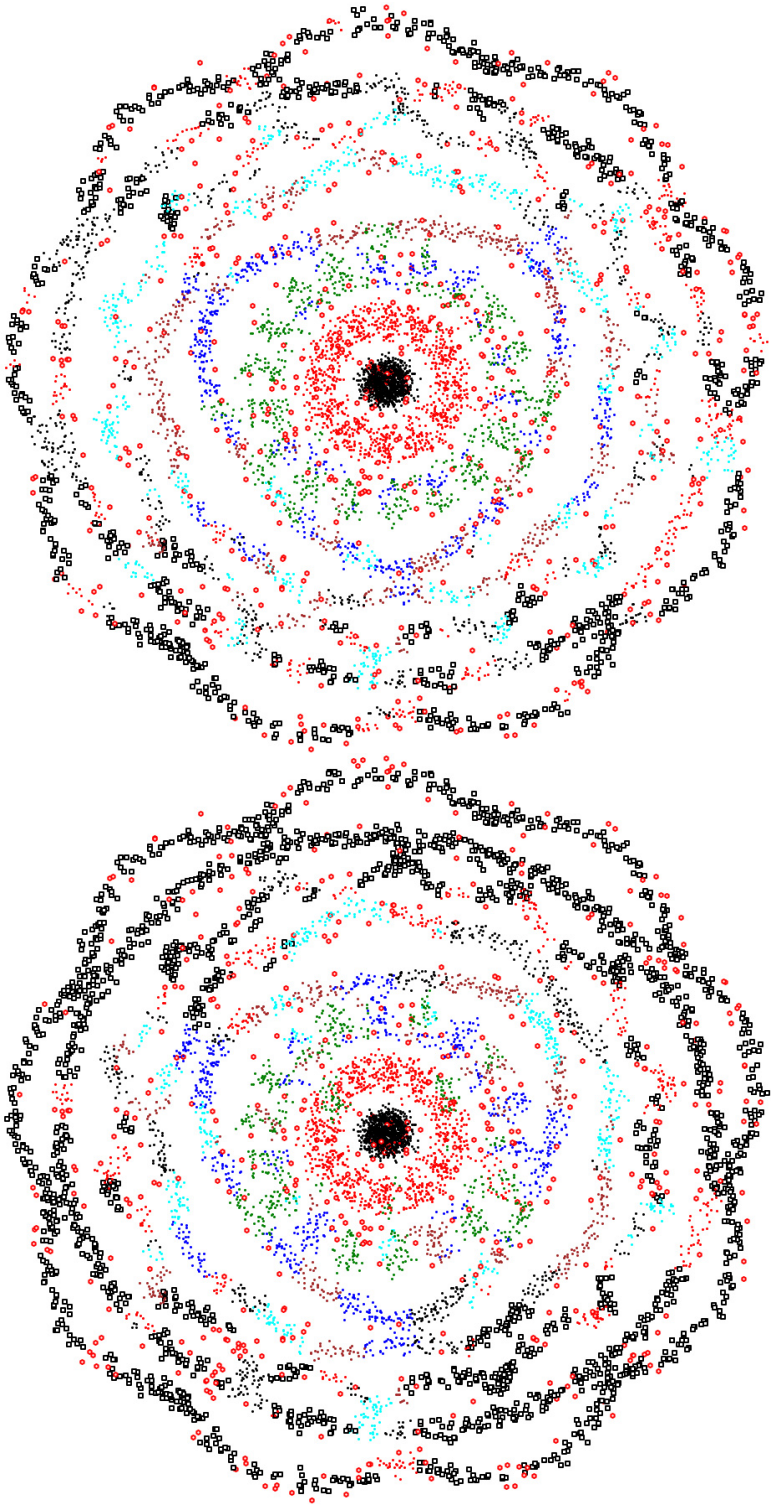

Figure 6: The parallelized DBSCAN performance in the planar circular dataset worst case (Figure 4) with the dataset division into 50 (above) and 25 (below) intervals (subsets).

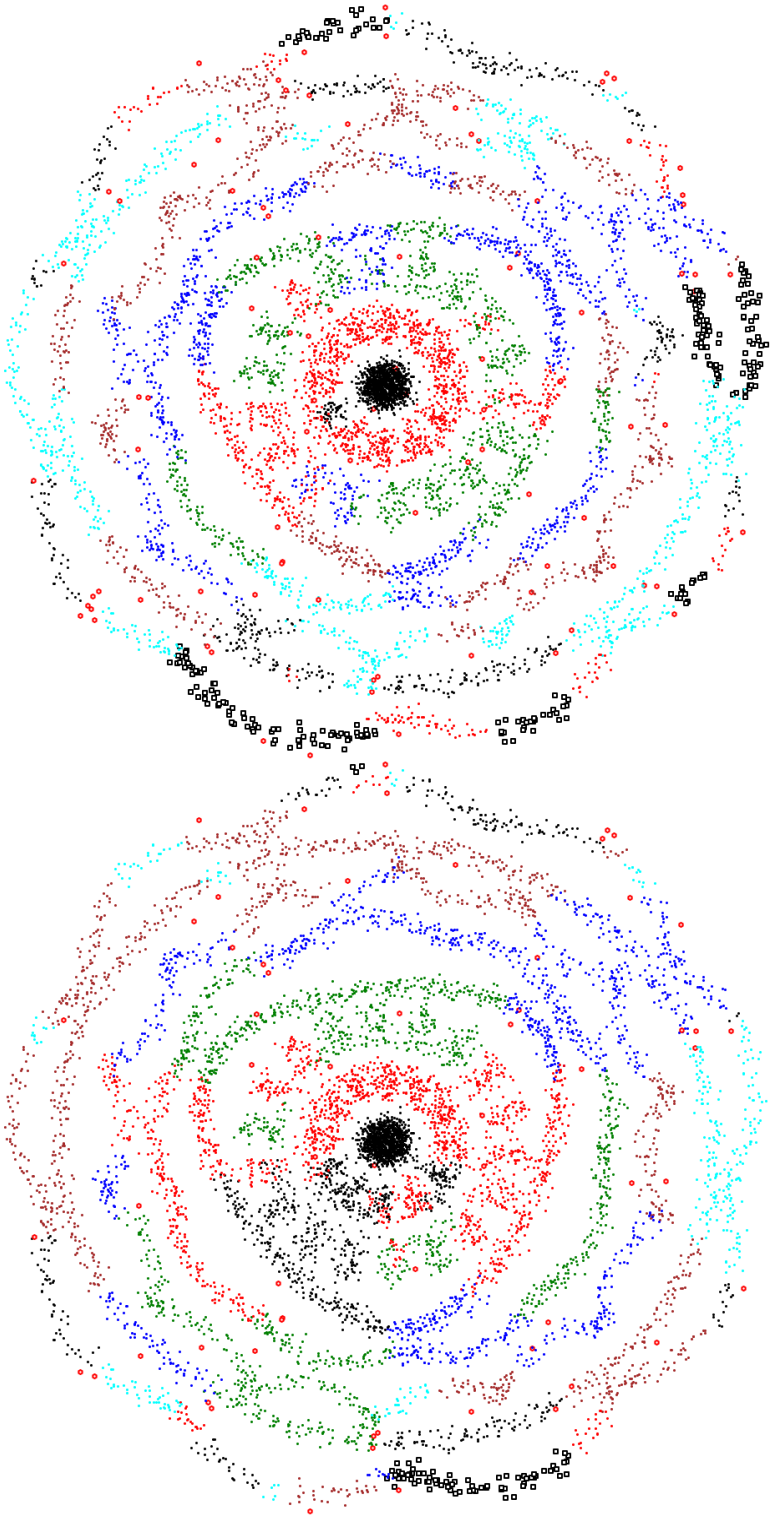

Figure 7: The parallelized DBSCAN performance in the planar circular dataset worst case (Figure 4) with the dataset division into 20 (above) and 10 (below) intervals (subsets).

$h_m = -1$ if $\mu < 0$; $g_m = 1$ if $\lambda > 0$ and $g_m = -1$ if $\lambda < 0$; c_1 is varied between 0.1 and 0.5 with a step of 0.1. Three-dimensional serpentine-like clusters form a developing dataset whose visualization is far much more complicated than that of the planar circular dataset (Figure 8).

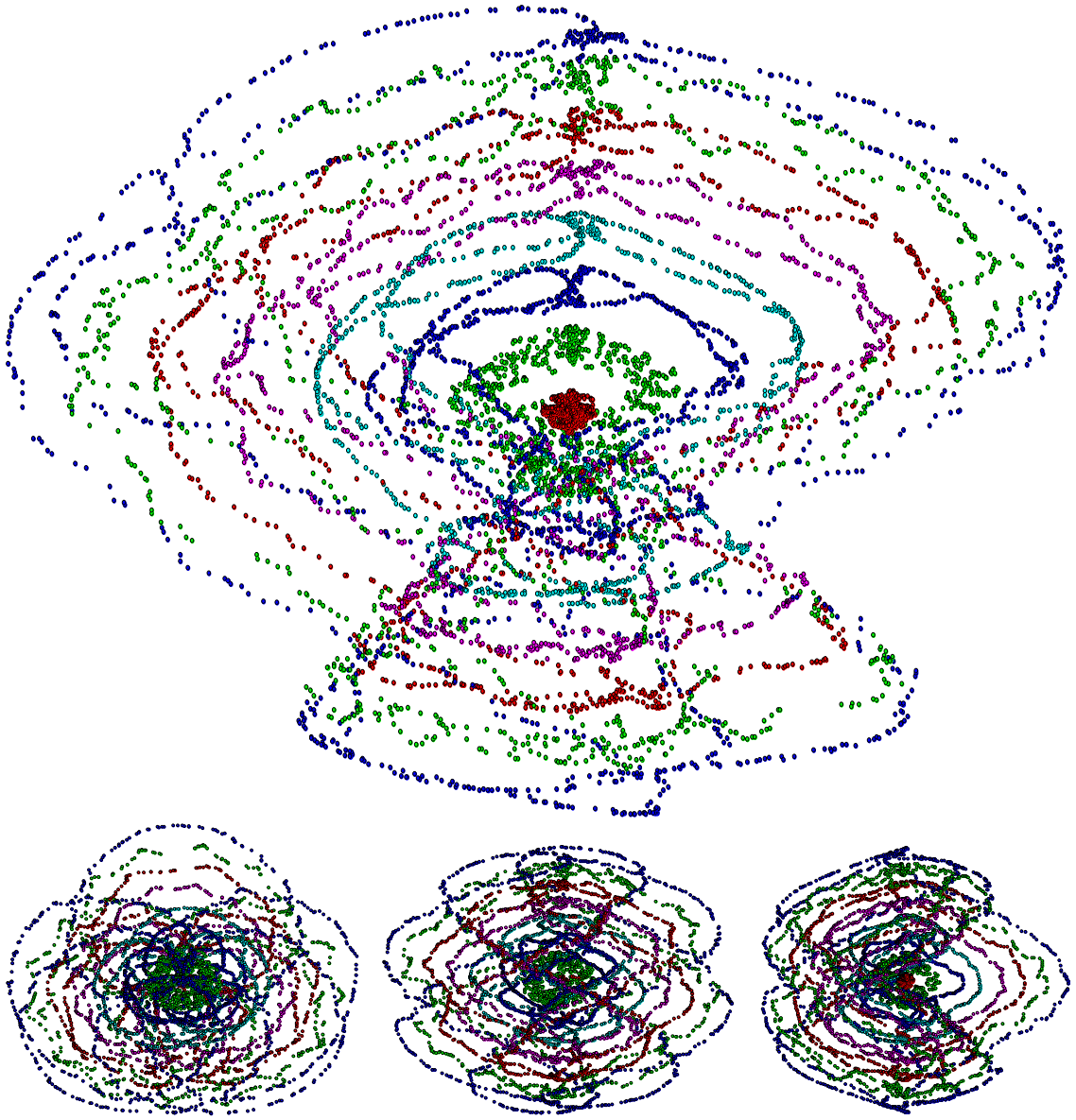

Figure 8: An example of the three-dimensional dataset comprised by 8 serpentine-like clusters (each cluster contains 1000 points); the xy, xz, and yz views are shown below exemplifying that such datasets bear distinct features of circular datasets.

The three-dimensional serpentine dataset by (14) — (16), (1), (13), (17), (18) is generated with three to eight clusters, $M = \overline{3,8}$, for five versions of noise magnitude,

$$c_1 \in \{0.1n\}_{n=1}^{5},$$

where each cluster contains 1000 to 10000 points with a step of 1000 points,

$$L \in \{1000k\}_{k=1}^{10}.$$

The remaining parameters of the three-dimensional serpentine dataset and its processing including (6) — (9) are the same as for the planar circular dataset.

Presented in Table 5, comparative computation time statistics for the three-dimensional serpentine dataset does not resemble that in Tables 1 and 2 for the planar datasets. Here, the parallelized DBSCAN speedup minimum is, prudently speaking, above 40 times and the speedup maximum exceeds 600 times, although standard deviations are higher than those for the planar datasets. On overall average, the parallelized DBSCAN over three-dimensional serpentine datasets is 342 times faster.

Table 5

Comparative computation time statistics for the three-dimensional serpentine dataset

Statistics	Minimum	Average	Maximum	Standard deviation
$\tau_{\text{noise}}(N, M)$	43.01525	304.0213	1136.548	218.268
$\tau_{\text{size}}(M, c_1)$	218.7431	338.8232	679.8032	116.2364
$\tau_M(N, c_1)$	44.25815	309.3223	953.4824	211.2102

Eventually presented in Table 6, comparative performance statistics for the three-dimensional serpentine dataset again confirms the advantage of the parallelized DBSCAN algorithm, but if the best subset size (best interval) is selected. The parallelized DBSCAN loses points at a higher rate (roughly, 12.5 %) than the original DBSCAN does (roughly, 8.6 %), but the best-interval parallelized DBSCAN misses points at a rate of below 4.8 %. The original DBSCAN rarely "sees" outliers having their average rate below 0.1 % and its maximum below 12.5 %, but it "sees" non-existing clusters at a rate of 2.97 % and confuses clusters at a rate of 5.57 %. The respective rates of non-existing clusters and cluster confusion for the parallelized DBSCAN are 5.37 % and 4.39 %, whereas they drop to 2.086 % and 2.065 % with the best-interval parallelized DBSCAN.

The worst three-dimensional case is in Figure 8, where 76.725 % of points are missed by the best-interval parallelized DBSCAN with the dataset division into 100 subsets. Meanwhile, the original DBSCAN performs at a rate of 24.25 % on this dataset. It is worth noting that in this case the noise magnitude is minimal. It appears to be paradoxical, but at greater noise magnitudes the best-interval parallelized DBSCAN performs even worse on some instances of the three-dimensional dataset with the maximum of clusters. The particular dataset in Figure 8 turns out to be the worst case due to the original DBSCAN performs far better on it (it could be even acceptable in some practical situations), and thus the difference between the original and best-interval parallelized DBSCAN algorithms is paradoxically huge.

Table 6
Comparative performance statistics for the three-dimensional serpentine dataset

Statistics		Minimum	Average	Maximum	Standard deviation
ε^*	DBSCAN	0.31	2.35159	4.86	0.72883
	Five intervals	0.01	1.51307	5.4	0.7421
	Best interval	0.29	1.73727	5.26	0.80428
$q^*_{\min}$	DBSCAN	3	3.51867	23	1.98233
	Five intervals	3	3.0972	14	0.65372
	Best interval	3	3.03367	9	0.29922
N_{outliers}	DBSCAN	0	0.05302	12.17857	0.29464
	Five intervals	0	2.75551	100	10.64053
	Best interval	0	0.62252	56.16667	2.80441
N_{other}	DBSCAN	0	2.97478	71.8	8.54771
	Five intervals	0	5.37361	67.63	8.39236
	Best interval	0	2.08576	51.025	4.12471
N_{confused}	DBSCAN	0	5.56964	58.4	10.04399
	Five intervals	0	4.38729	74.45714	6.41309
	Best interval	0	2.06541	54.625	4.71045
ρ	DBSCAN	0	8.59744	75.0875	14.35587
	Five intervals	0	12.51642	100	18.76093
	Best interval	0	4.77369	76.725	9.18129

5. Discussion and conclusion

Compared to the performance of the original DBSCAN algorithm, the parallelized DBSCAN performs more accurately being extremely fast. The latter is straightforwardly inferred from Tables 1, 3, 5, where the average speedup exceeds 80 times for planar datasets and exceeds 300 times for three-dimensional datasets. The gain in accuracy is not that unambiguous. While the parallelized DBSCAN outperforms the original DBSCAN on planar

datasets (Tables 2 and 4), even without selecting the best size of the subset (called the best interval due to planarity), it underperforms on three-dimensional datasets without selecting the best-interval result. Only the best-interval parallelized DBSCAN outperforms the original DBSCAN on three-dimensional datasets. It is likely reasoned by a higher complexity of the structure of such datasets (see Figure 8) compared to the structure of planar datasets (see Figures 1 — 4). In other words, the DBSCAN parallelization by spatial dataset division and hyperparameter adjustment is less efficient on datasets with more complex structure. However, the series of computational experiments has exposed a possibility of the parallelized DBSCAN underperformance for simple-structured datasets also (see Figure 3 with the worst case of the planar serpentine dataset, although its structure is far much simpler than the structure of planar circular dataset in Figure 4).

The growing number of clusters may additionally affect the parallelized DBSCAN performance. Nevertheless, as the dataset size increases and the number of clusters increases, one by one or together, the parallelized DBSCAN outperforms the original DBSCAN more. The gain in accuracy ranges from 4.9072 to 276.8096 times for the planar serpentine dataset, when an averaged over noise ratio of the original DBSCAN accuracy to the parallelized DBSCAN accuracy is considered as a function of the dataset size and the number of clusters. The accuracy gain drastically drops for the planar circular dataset ranging from 1.4636 to 5.5163 times. For the three-dimensional serpentine dataset, the accuracy gain ranges from 0.0004 to 10.9631, i. e. it is below 1 for three and four clusters (it is just 0.1007 and 0.3179 for three and four clusters, respectively). For five clusters and more, along with the dataset size increasing, the parallelized DBSCAN outperforms the original DBSCAN. The poorer performance on the fewest number of clusters is an obvious limitation of the parallelized DBSCAN.

Another limitation is the poorer performance on datasets with circularity. Dividing into intervals is uncommon for circular datasets, whose natural division would be circular-sector, especially for three-dimensional datasets. Moreover, it is hard to divide so that each cluster would contain (approximately) the same number of points within every subset. In addition, the DBSCAN parallelization efficiency is impossible to estimate without known ground truth labels.

The DBSCAN parallelization efficiency expectedly deteriorates when density is changeable. The dataset in Figure 3 is seemingly the case, where each serpentine has significantly thinner and thicker intervals observable via zoom-in. This is the most common drawback of density-based clustering methods, although they all, and the DBSCAN algorithm in particular, are intended to handle varying densities of points. However, the datasets used for the computational experiments have density-modulated regions (i. e., varying density of regions of changeable densities of points), which have served as a close-to-the-worst-case scenario in order to reveal weaknesses of the suggested DBSCAN parallelization.

Overall, it is a promising approach to speed up arbitrary-shape clustering without losing in accuracy. Posed as a local optimization versus global optimization, the parallelized DBSCAN performs well on serpentine datasets but not limited only to them. The hyperparameters, neighborhood radius and minimum-neighbors number, are gradually adjusted with some steps, which are tunable also (basically, the neighborhood

radius increment step). Apart from the subsets of the divided dataset, the suggested modification of the DBSCAN algorithm can be run in parallel for multiple labeled subset sizes, as well as for a few versions of the neighborhood radius increment step.

References

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

[2] R. J. G. B. Campello, D. Moulavi, J. Sander, Density-based clustering based on hierarchical density estimates, in: J. Pei, V. S. Tseng, L. Cao, H. Motoda (Eds.), Advances in Knowledge Discovery and Data Mining, Vol. 7819, Springer Berlin Heidelberg, 2013, pp. 160–172. doi:10.1007/978-3-642-37456-2_14.

[3] S. Weng, Z. Fan, J. Gou, A fast DBSCAN algorithm using a bi-directional HNSW index structure for big data, International Journal of Machine Learning and Cybernetics 15 (2024) 3471–3494. doi:10.1007/s13042-024-02104-8.

[4] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, X. Xu, DBSCAN Revisited, revisited: Why and how you should (still) use DBSCAN, ACM Transactions on Database Systems 42 (3) (2017) 1–21. doi:10.1145/3068335.

[5] J. Sander, Generalized Density-Based Clustering for Spatial Data Mining, Herbert Utz Verlag, München, 1998.

[6] X. Yang, X. Zhou, B. Wan et al., Load spectra extrapolation by bandwidth-optimized kernel density estimation based on DBSCAN algorithm, Journal of Vibration Engineering & Technologies 12 (2024) 1445–1456. doi:10.1007/s42417-023-00919-3.

[7] J. Sander, M. Ester, H.-P. Kriegel, X. Xu, Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications, Data Mining and Knowledge Discovery 2 (2) (1998) 169–194. doi:10.1023/A:1009745219419.

[8] W. Zhang, An improved DBSCAN algorithm for hazard recognition of obstacles in unmanned scenes, Soft Computing 27 (2023) 18585–18604. doi:10.1007/s00500-023-09319-x.

[9] V. V. Romanuke, Speedup of the *k*-means algorithm for partitioning large datasets of flat points by a preliminary partition and selecting initial centroids, Applied Computer Systems 28 (1) (2023) 1–12. doi:10.2478/acss-2023-0001.

[10] V. V. Romanuke, S. V. Merinova, H. A. Yehoshyna, Optimized centroid-based clustering of dense nearly-square point clouds by the hexagonal pattern, Electrical, Control and Communication Engineering 19 (1) (2023) 29–39. doi:10.2478/ecce-2023-0005.

[11] S. Xiao, Z. Zhou, Z. Chen, Y. Qi, Bus station location selection method based on DBSCAN-DPC clustering algorithm, in: Y. Qu, M. Gu, Y. Niu, W. Fu (Eds.), Proceedings of 3rd 2023 International Conference on Autonomous Unmanned Systems (3rd ICAUS 2023), ICAUS 2023, Lecture Notes in Electrical Engineering, vol. 1177, Springer, Singapore, 2024. doi:10.1007/978-981-97-1103-1_13.

[12] Y. Mao, D. S. Mwakapesa, Y. Li et al., Assessment of landslide susceptibility using DBSCAN-AHD and LD-EV methods, Journal of Mountain Science 19 (2022) 184–197. doi:10.1007/s11629-020-6491-7.

[13] Z. Qi, H. Wang, Z. Dong, Density-Based Clustering for Incomplete Data, in: Dirty Data Processing for Machine Learning, Springer, Singapore, 2024. doi:10.1007/978-981-99-7657-7_5.
[14] A. Starczewski, A Novel Approach to Determining the Radius of the Neighborhood Required for the DBSCAN Algorithm, in: L. Rutkowski, R. Scherer, M. Korytkowski, W. Pedrycz, R. Tadeusiewicz, J. M. Zurada (Eds.), Artificial Intelligence and Soft Computing, ICAISC 2021, Lecture Notes in Computer Science, vol. 12854, Springer, Cham, 2021. doi:10.1007/978-3-030-87986-0_32.
[15] J. A. Hartigan, M. A. Wong, Algorithm AS 136: A *k*-means clustering algorithm, Journal of the Royal Statistical Society, Ser. C 28 (1) (1979) 100–108. doi:10.2307/2346830.
[16] M. E. Celebi, H. A. Kingravi, P. A. Vela, A comparative study of efficient initialization methods for the *k*-means clustering algorithm, Expert Systems with Applications 40 (1) (2013) 200–210. doi:10.1016/j.eswa.2012.07.021.
[17] R. J. G. B. Campello, D. Moulavi, A. Zimek, J. Sander, Hierarchical density estimates for data clustering, visualization, and outlier detection, ACM Transactions on Knowledge Discovery from Data 10 (1) (2015) 1–51. doi:10.1145/2733381.
[18] H.-P. Kriegel, P. Kröger, J. Sander, A. Zimek, Density-based clustering, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 1 (3) (2011) 231–240. doi:10.1002/widm.30.
[19] Y. Zuo, Z. Hu, S. Yuan et al., Identification of convective and stratiform clouds based on the improved DBSCAN clustering algorithm, Advances in Atmospheric Sciences 39 (2022) 2203–2212. doi:10.1007/s00376-021-1223-7.
[20] G. Habib, S. Qureshi, Convolutional neural networks (CNN) and DBSCAN clustering for SARs-CoV challenges: complete deep learning solution, in: D. Gupta, A. Khanna, S. Bhattacharyya, A. E. Hassanien, S. Anand, A. Jaiswal (Eds.), International Conference on Innovative Computing and Communications, Lecture Notes in Networks and Systems, vol. 471, Springer, Singapore, 2023. doi:10.1007/978-981-19-2535-1_35.
[21] M. Ankerst, M. M. Breunig, H.-P. Kriegel, J. Sander, OPTICS: Ordering points to identify the clustering structure, in: ACM SIGMOD International Conference on Management of Data, ACM Press, 1999, pp. 49–60.
[22] R. J. G. B. Campello, D. Moulavi, A. Zimek, J. Sander, A framework for semi-supervised and unsupervised optimal extraction of clusters from hierarchies, Data Mining and Knowledge Discovery 27 (3) (2013) 344. doi:10.1007/s10618-013-0311-4.
[23] Z. F. Wang, P. Y. Yuan, Z. Y. Cao et al., Feature reduction of unbalanced data classification based on density clustering, Computing 106 (2024) 29–55. doi:10.1007/s00607-023-01206-5.
[24] Z. Qi, H. Wang, Z. Dong, Density-Based Clustering for Incomplete Data, in: Dirty Data Processing for Machine Learning, Springer, Singapore, 2024. doi:10.1007/978-981-99-7657-7_5.
[25] H. Zhang, B. Liu, P. Cui, Y. Sun, Y. Yang, S. Guo, An outlier detection algorithm for electric power data based on DBSCAN and LOF, in: Q. Liu, X. Liu, L. Li, H. Zhou, H. H. Zhao (Eds.), Proceedings of the 9th International Conference on Computer Engineering and Networks, Advances in Intelligent Systems and Computing, vol. 1143, Springer, Singapore, 2021. doi:10.1007/978-981-15-3753-0_110.

[26] D. Rangaprakash, T. Odemuyiwa, D. Narayana Dutt et al., Density-based clustering of static and dynamic functional MRI connectivity features obtained from subjects with cognitive impairment, Brain Informatics 7 (2020) 19. doi:10.1186/s40708-020-00120-2.
[27] H. T. Nguyen, T. H. Phan, L. T. T. Pham et al., Clustering-based visualizations for diagnosing diseases on metagenomic data, Signal, Image and Video Processing 18 (2024) 5685–5699. doi:10.1007/s11760-024-03264-4.
[28] P. Sarang, Density-Based Clustering, in: Thinking Data Science, The Springer Series in Applied Machine Learning, Springer, Cham, 2023. doi:10.1007/978-3-031-02363-7_12.
[29] Y. R. Pan, Y. H. Xia, L. J. Long et al., Power-line extraction and modelling from 3D point clouds data based on K-D tree DBSCAN algorithm, Journal of Electrical Engineering & Technology 19 (2024) 3587–3597. doi:10.1007/s42835-023-01641-6.
[30] C. L. Valenzuela, A. J. Jones, Evolutionary divide and conquer (I): A novel genetic approach to the TSP, Evolutionary Computation 1 (4) (1993) 313–333. doi:10.1162/evco.1993.1.4.313.
[31] V. V. Romanuke, Traveling salesman problem parallelization by solving clustered subproblems, Foundations of Computing and Decision Sciences 48 (4) (2023) 453–481. doi:10.2478/fcds-2023-0020.
[32] V. V. Romanuke, Deep clustering of the traveling salesman problem to parallelize its solution, Computers & Operations Research 165 (2024) 106548. doi:10.1016/j.cor.2024.106548.
[33] C. Cortes, V. Vapnik, Support-vector networks, Machine Learning 20 (3) (1995) 273–297. doi:10.1007/BF00994018.
[34] A. Ben-Hur, D. Horn, H. Siegelmann, V. Vapnik, Support vector clustering, Journal of Machine Learning Research 2 (2001) 125–137.
[35] V. V. Romanuke, Optimization of a dataset for a machine learning task by clustering and selecting closest-to-the-centroid objects, Herald of Khmelnytskyi national university. Technical sciences 6 (1) (2018) 263–265.
[36] V. V. Romanuke, Optimal partitioning of an initial dataset into subdatasets to be clustered for getting rid off the dataset superfluities for a machine learning task, Herald of Khmelnytskyi national university. Technical sciences 6 (2) (2018) 213–215.
[37] V. V. Romanuke, Random centroid initialization for improving centroid-based clustering, Decision Making: Applications in Management and Engineering 6 (2) (2023) 734–746. doi:10.31181/dmame622023742.
[38] V. V. Romanuke, Parallelization of the traveling salesman problem by clustering its nodes and finding the best route passing through the centroids, Applied Computer Systems 28 (2) (2023) 189–202. doi:10.2478/acss-2023-0019.
[39] K. C. Bhupathi, H. Ferdowsi, Sharp curve detection of autonomous vehicles using DBSCAN and augmented sliding window techniques, International Journal of Intelligent Transportation Systems Research 20 (2022) 651–671. doi:10.1007/s13177-022-00317-1.
[40] Y. Zack, Cluster analysis for multidimensional objects in fuzzy data conditions, System Research and Information Technologies 2 (2021) 18–34. doi:10.20535/SRIT.2308-8893.2021.2.02.