Applied Computer Systems
ISSN 2255-8691 (online)
ISSN 2255-8683 (print)
June 2024, vol. 29, no. 1, pp. 14–23
https://doi.org/10.2478/acss-2024-0003
https://content.sciendo.com

DBSCAN Speedup for Time-Serpentine Datasets

Vadim Romanuke*
Polish Naval Academy, Gdynia, Poland

***Abstract* – An approach to speed up the DBSCAN algorithm is suggested. The planar clusters to be revealed are assumed to be tightly packed and correlated constituting, thus, a serpentine dataset developing rightwards or leftwards as time goes on. The dataset is initially divided into a few sub-datasets along the time axis, whereupon the best neighbourhood radius is determined over the first sub-dataset and the standard DBSCAN algorithm is run over all the sub-datasets by the best neighbourhood radius. To find the best neighbourhood radius, it is necessary to know ground truth cluster labels of points within a region. The factual speedup registered in a series of 80 000 dataset computational simulations ranges from 5.0365 to 724.7633 having a trend to increase as the dataset size increases.**

***Keywords* – Clustering, DBSCAN, large dataset, serpentine cluster, speedup.**

I. The DBSCAN Algorithm Performance

The DBSCAN algorithm is capable of revealing clusters of arbitrary shape by domain minimal knowledge [1], [2]. Fundamentally, DBSCAN groups points using the conception of ε-neighbourhood of a point and a specified minimum number of neighbouring points $q_{\min}$ [3], [4]. This is further supplemented with the conceptions of core points, directly reachable points, non-directly reachable points, and outliers [2], [5], [6]. Furthermore, a symmetric notion of density-connectedness is added to rectify the non-symmetric relation of the reachability [3], [7]. DBSCAN, thus, defines the density by the neighbourhood radius ε and number $q_{\min}$, where a cluster contains at least $q_{\min}$ points. The ε-neighbourhood is defined as a set of points falling within radius ε by using a preliminarily taken distance function [3], [4], [8], [9].

Contrary to other commonly used clustering algorithms like, e. g., *k*-means [4], [10], [11], the DBSCAN algorithm performs much better in identifying non-globular shapes and revealing outliers [12], [13]. Along with incomparably higher accuracy on such datasets, DBSCAN performs fast enough. Thus, for N points in a dataset, the DBSCAN average runtime complexity is $O(N\log N)$ [14], [15]. However, if a dataset contains regions whose density varies too much, DBSCAN hyper-parameters ε and $q_{\min}$ require finer tuning [16], [17]. Moreover, the distance function defining the ε-neighbourhood becomes another hyper-parameter, which may heavily affect the DBSCAN accuracy [18], [19]. Furthermore, volatile-density regions provoke worst-case scenarios of the DBSCAN runtime whose quadratic complexity $O(N^2)$ is also possible [20], [21], not including time spent for tuning the DBSCAN hyper-parameters.

DBSCAN performs worse on larger datasets [22]. Finding an appropriate value for ε takes more time and computational resources for higher-dimensional data. While OPTICS, another clustering algorithm, mostly rectifies the mentioned drawbacks of DBSCAN, its performance on larger datasets is far poorer because the OPTICS algorithm is mainly intended for hierarchical clustering [23], [24]. In addition, OPTICS on average is 1.6 times slower than DBSCAN [25], [26].

II. Goal and Tasks to be Accomplished

Time-developing clusters are an example of non-globular shapes, where points occupy new regions rightwards or leftwards as time goes on. A time series can be a particular case of a planar dataset of time-developing clusters, but generally the latter are generated by digital traces of a few objects tracked by an observer [27], [28]. Visually a dataset of such clusters reminds a bunch of serpentine traces on a two-dimensional map, so it can be called a time-serpentine dataset [29]. The number of clusters tracked is known, but the number of their registered positions grows severely due to noise that breaks a registered position into a few ones [30], [31].

As the DBSCAN algorithm performs poorly on large datasets, the goal is to study a possibility of speeding up the DBSCAN performance by not decreasing its accuracy with respect to time-serpentine datasets. To achieve the goal, the following four tasks are to be accomplished:

1. To suggest an approach that could possibly speed up the DBSCAN algorithm clustering of large time-serpentine datasets of planar points.
2. To study the performance of the suggested DBSCAN update on time-serpentine datasets.
3. To discuss the possible speedup, if any, of the updated DBSCAN algorithm, and rank limitations of its application.
4. To conclude on the scientific and practical contribution to applied computer science and outline a possibility of further research.

III. Hyper-parameter Tuning

An approach that could possibly speed up the DBSCAN algorithm clustering must be based on two items:

* Corresponding author's e-mail: romanukevadimv@gmail.com
Article received 2024-04-12; accepted 2024-07-10

1. Using the best values of the DBSCAN hyper-parameters.
2. Clustering in parallel by preliminarily partitioning the dataset into sub-datasets.

Usually the most rational value of $q_{\min}$ is 3 and the distance function is Euclidean [3], [32], [33]. Therefore, within initial approximation, it remains to find a quasi-optimal value of the neighbourhood radius. It is assumed that a large serpentine dataset is of either non-sparsely scattered clusters or tightly connected clusters, so it is possible to divide the dataset into multiple sub-datasets in order to apply the standard DBSCAN algorithm to every sub-dataset separately (in parallel, if parallel processor cores is available). This resembles the divide-and-conquer approach, according to which a large combinatorial optimization problem is itself clustered into sub-problems, whereupon they are solved independently [34], [35]. For instance, this has been successfully applied to solving large traveling salesman problems whose nodes are scattered within a globular shape [36], [37]. However, a serpentine dataset of time-developing clusters cannot be clustered in that way. A part of tightly connected points is taken instead. Points in this part, which is a sub-dataset of a reasonable size, should represent all the clusters having approximately the same number of representatives in each cluster.

Hence, the dataset is divided into d sub-datasets, and the very first sub-dataset is repeatedly clustered by the standard DBSCAN algorithm within a range of the neighbourhood radius. Obviously, the minimum $\varepsilon_{\min}$ and maximum $\varepsilon_{\max}$ values in this range depend on the scale of points. So does the step $\Delta\varepsilon$ by which the neighbourhood radius is incremented. The selection of these three values stands out as an additional task that requires some experience and knowledge about the dataset scale [38], [39].

The neighbourhood radius is a variable to the performance accuracy function represented as a percentage of points incorrectly assigned (generally speaking, missed) upon clustering. This function $\rho_{\text{miss}}(\varepsilon)$ is studied on the neighbourhood radius interval $[\varepsilon_{\min}; \varepsilon_{\max}]$. Value $\rho_{\text{miss}}(\varepsilon)$ is decomposed into three categories of the missed points: outliers whose percentage is $\rho_{\text{out}}(\varepsilon)$, points incorrectly assigned to non-existing (other) clusters whose percentage is $\rho_{\text{oth}}(\varepsilon)$, and points incorrectly assigned to existing clusters but their labels are factually incorrect (confused) at a percentage rate of $\rho_{\text{conf}}(\varepsilon)$. Thus,

$$\rho_{\text{miss}}(\varepsilon) = \rho_{\text{out}}(\varepsilon) + \rho_{\text{oth}}(\varepsilon) + \rho_{\text{conf}}(\varepsilon). \tag{1}$$

In general, the best neighbourhood radius is selected as

$$\varepsilon^* \in \min_{\varepsilon \in [\varepsilon_{\min}; \varepsilon_{\max}]} \rho_{\text{miss}}(\varepsilon), \tag{2}$$

resulting in the minimum possible percentage of missed points

$$\rho^*_{\text{miss}} = \rho_{\text{miss}}(\varepsilon^*), \tag{3}$$

which is the best achievable accuracy. Nevertheless, it is worth noting that the solution of problem (2) may not imply that percentages

$$\rho^*_{\text{out}} = \rho_{\text{out}}(\varepsilon^*), \tag{4}$$

$$\rho^*_{\text{oth}} = \rho_{\text{oth}}(\varepsilon^*), \tag{5}$$

$$\rho^*_{\text{conf}} = \rho_{\text{conf}}(\varepsilon^*) \tag{6}$$

are simultaneously minimal along with (3). Besides, the best achievable accuracy may heavily depend on how $\Delta\varepsilon$ is selected [40], [41].

Having found the best neighbourhood radius, at which the primarily taken first sub-dataset is already clustered optimally with the accuracy of (3), the remaining $d - 1$ sub-datasets are clustered by ε^*. The overall accuracy is the sum of percentages (3) taken over the d sub-datasets. The percentages of the components of the overall accuracy decomposed into (4)–(6) are calculated in the same way.

Obviously, such a hyper-parameter tuning is possible only by knowing ground truth cluster labels of the first sub-dataset points. Values $\varepsilon_{\min}$, $\varepsilon_{\max}$, $\Delta\varepsilon$ are the neighbourhood radius tuning parameters (which subsequently might be called the tuning meta-parameters).

IV. Performance on Serpentine Datasets

The performance of the suggested DBSCAN update (or, the tuned DBSCAN, in other words) on time-serpentine datasets is studied over values

$$\varepsilon^*,\ \rho^*_{\text{out}},\ \rho^*_{\text{oth}},\ \rho^*_{\text{conf}},\ \rho^*_{\text{miss}}. \tag{7}$$

The serpentine dataset size S is varied from $10^3 \cdot C$ to $10^4 \cdot C$ points with a step of $10^3 \cdot C$ points, where the factual number of clusters C is varied from three to 18 clusters. A serpentine dataset point

$$[x_k \quad y_k] \in \quad ^2$$

is randomly generated using values

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

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

$$c = \overline{1, C} \text{ and } t = \overline{1, T},\ k = t + (c-1) \cdot T,\ k = \overline{1, S} \tag{8}$$

and $S = C \cdot T$:

$$x_k = \frac{100\pi}{T-1} \cdot (t-1) + 3\pi c + \alpha\xi_{ct} \tag{9}$$

by (8), where α is randomly chosen from a set of numbers

$$\left\{0.1 + \frac{j}{1110}\right\}_{j=0}^{999}. \tag{10}$$

Values η and μ of two normally distributed random variables with zero mean and unit variance are additionally used to impart more volatility to the dataset density along vertical component

$$y_k = (\beta_{1ct}\sin(\omega_1 x_k) + \beta_{2ct}\cos(\omega_2 x_k))\cdot e^{\lambda x_k} + \\ +\gamma_1 x_k + \gamma_2 \zeta_{ct} - 4\cdot(c-1) \quad (11)$$

by (9), (8), and

$$\beta_{1ct} = \beta_{01ct} + 0.125\vartheta_{ct}, \quad (12)$$

$$\beta_{2ct} = \beta_{02ct} + 0.125\vartheta_{ct}. \quad (13)$$

In (12), (13), both β_{01ct} and β_{02ct} are independently randomly chosen from a set of numbers

$$\left\{3 + \frac{j}{999}\right\}_{j=0}^{999}. \quad (14)$$

In (11), both ω_1 and ω_2 are independently randomly chosen from a set of numbers

$$\left\{0.01 + \frac{j}{11100}\right\}_{j=0}^{999}; \quad (15)$$

exponential growth coefficient λ is chosen randomly from a set of numbers

$$\left\{0.0005 + \frac{j}{222000}\right\}_{j=0}^{999} \quad (16)$$

and it is set negative if $\eta < 0$; coefficient γ_1 is chosen independently randomly from set (16) and it is set negative if $\mu < 0$; coefficient γ_2 is non-randomly set to values

$$\{0.1, 0.2, 0.3, 0.4, 0.5\} \quad (17)$$

by the order in set (17). Values in (17) constitute a set of five versions of noise magnitude. An example of the serpentine dataset generated by (9), (11), (8), (10), (12)–(16) at the noise maximum is presented in Fig. 1. Another example, at the same noise magnitude, is presented in Fig. 2, where the number of clusters is six times greater. It is worth noting that the horizontal scaling of these scatter graphs is the same, and the vertical scaling is maintained at the same aspect ratio. Thus, the seeming difference in the "thickness" of the clusters is an illusion caused by a vaster domain of the dataset with 18 clusters. A subtle peculiarity herein (clearly seen in Fig. 1, although less clearly observable in Fig. 2) is the cluster density, which is lower at cluster boundaries as points within every cluster interior are located more compact. A subtler peculiarity is that the "thickness" of the seemingly "thinner" 18 clusters in Fig. 2 varies far more than that in Fig. 1.

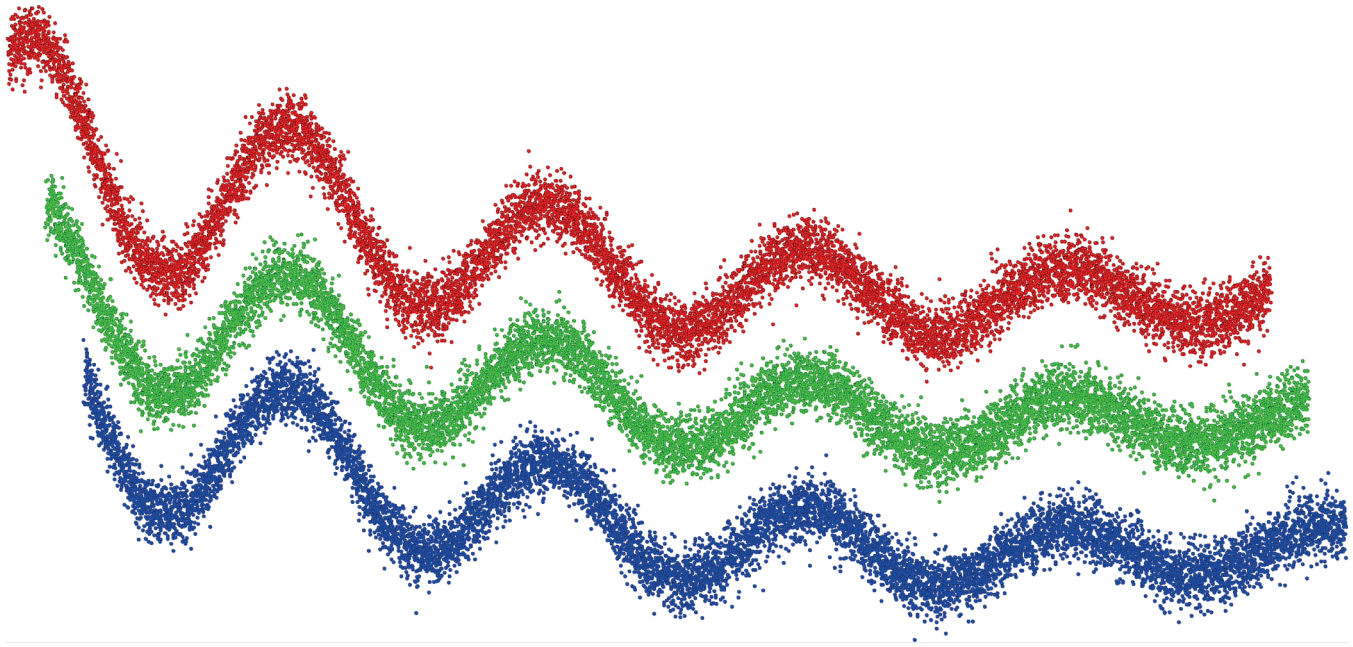

Fig. 1. An example of the serpentine dataset of 30 000 points within three clusters, 10 000 points in each cluster. The noise is at maximum. Visually, the clusters are easily separated by the cluster density, but it is lower at cluster boundaries as points within every cluster interior are located more compact.

A specificity of the datasets generated by (9), (11), (8), (10), (12)–(17) is that every next cluster whose points located vertically lower is shifted to the right. This is intentionally done to prevent an almost simultaneous "start" of clusters of a time-serpentine dataset. Consequently, the lower cluster "ends later" on the right. Besides, clusters in a dataset resemble each other to some extent. For instance, the meandering clusters in Fig. 1 appear to be just replicas of a noised pattern, apart from the small starting part of the upper cluster. Nevertheless, replication is not regular – in the upper part of the dataset in Fig. 2, for instance, the fourth cluster from above has an almost vertically-stable region quite dissimilar to the same regions of the clusters above (see Fig. 3).

Figure 3 also shows an example of that clusters severely vary in density, although it may be not apparent from the general view (like that in Fig. 2). In vertical direction, clusters tend to be the "thinnest". Contrary to that, clusters tend to be the "thickest" in horizontal direction.

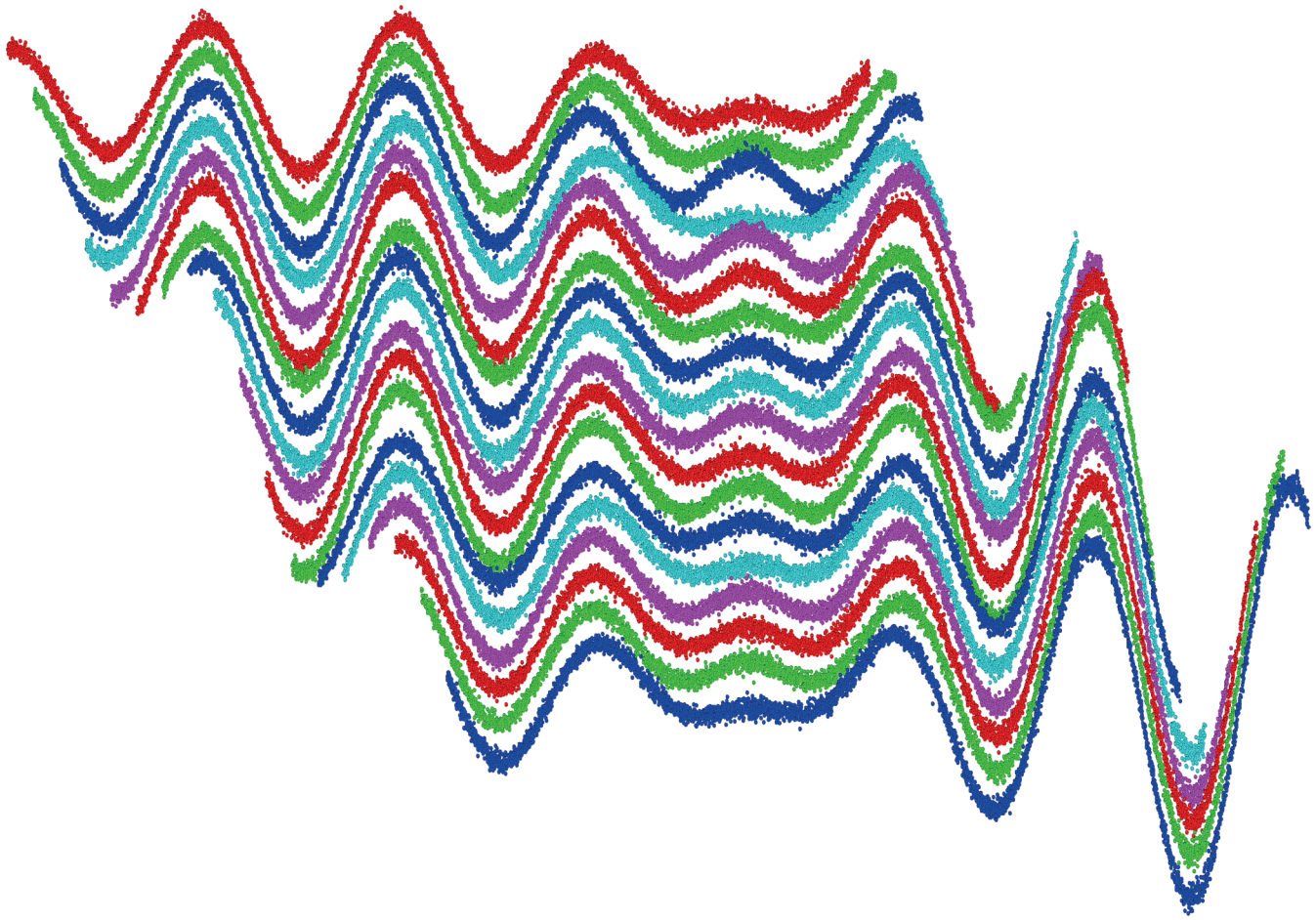

Fig. 2. An example of the serpentine dataset of 90 000 points within 18 clusters, 5000 points in each cluster. Despite the noise is at maximum, the cluster density seemingly is not as much varying as that in Fig. 1, where the three clusters appear to be "thicker". Here the 18 clusters seem to be much "thinner" being still well-separated visually. In reality, however, the "thickness" of the clusters varies far more than that in Fig. 1 (see Fig. 3).

A more obvious feature of the serpentine datasets generated by (9), (11), (8), (10), (12)–(17) is a distinct synchronism of their clusters. The serpentine arcs nearly coincide, which is clearly seen in Figs. 1–3. Moreover, concavity and convexity of the serpentine arcs coincide exactly. This is intentionally done to make recognisability and distinguishability of clusters more difficult. This allows studying worst-case scenarios of the clustering algorithm performance by having greater gaps in accuracy and, likely, deliberate slowdown in runtime.

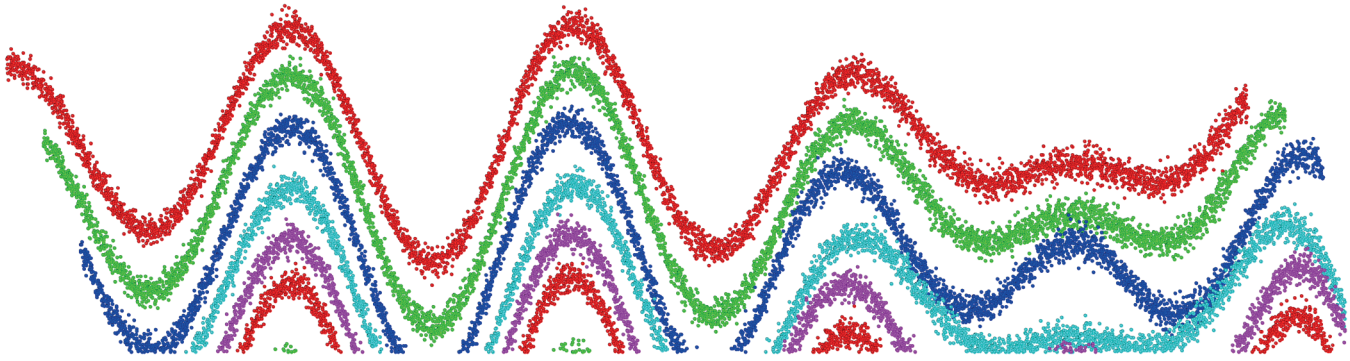

Fig. 3. A cut-off of a zoom-in on the upper part of the dataset in Fig. 2 (the same aspect ratio is maintained). The zoom-in shows the varying density of the clusters in more detail. The "thickness" of the clusters is near its minimum when the serpentine develops more in vertical direction. The "thickness" of the clusters is near its maximum at either the serpentine arcs or when the serpentine develops more in horizontal direction.

The serpentine dataset generation is repeated for 100 times, where the first sub-dataset with ground truth cluster labels is a tenth part of the whole dataset. The performance results are obtained and registered for the suggested DBSCAN update with the hyper-parameter tuning (briefly, 1-tuned DBSCAN), for which a quasi-optimal value of the neighbourhood radius is

determined by (2), where $\varepsilon_{min} = 0.01$, $\varepsilon_{max} = 5$, $\Delta\varepsilon = 0.1$. Then the standard DBSCAN algorithm is applied to the dataset with the determined quasi-optimal value of the neighbourhood radius ε^*. The third approach lies in determining the best neighbourhood radius for the whole dataset and the subsequent application of the standard DBSCAN algorithm (for this approach, it is designated as DBSCAN*). Every numerical result of the performance (both accuracy and runtime) is stored as a 10×16×5×100 array. The four-dimensional array is averaged over the fourth dimension (repetitions) and processed further.

The minimum, average, and maximum percentages of the missed points are presented in Table I for each of the three approaches. The minimum is expectedly 0, and the maximum reaches unacceptable rates, apart from the outliers. Nevertheless, the 1-tuned DBSCAN inaccuracy is about 27.1 %, which is twice as lower at the standard DBSCAN, whereas DBSCAN* is also almost 30 % worse.

TABLE I
PERCENTAGES OF THE MISSED POINTS BY CATEGORIES

Performance	Algorithm	Minimum	Average	Maximum	Standard deviation
ρ^*_{out}	1-tuned DBSCAN	0	0.1238	2.86	0.2704
	DBSCAN	0	0.0802	2.5775	0.212
	DBSCAN*	0	0.1328	2.5688	0.2568
ρ^*_{oth}	1-tuned DBSCAN	0	1.3677	54.9604	4.8603
	DBSCAN	0	3.0116	92.9647	12.3539
	DBSCAN*	0	3.5257	81.4321	10.4487
ρ^*_{conf}	1-tuned DBSCAN	0	25.6548	94.4444	25.4395
	DBSCAN	0	51.1173	94.8833	38.7091
	DBSCAN*	0	33.2214	94.4444	34.9281
ρ^*_{miss}	1-tuned DBSCAN	0	27.1463	94.4444	26.7576
	DBSCAN	0	54.209	99.7235	39.3447
	DBSCAN*	0	36.8799	94.4444	37.2358

The 1-tuned DBSCAN averaged inaccuracy ρ^*_{miss} is presented in Table II with respect to the cluster size. The minimum is expectedly 0, and the maximum reaches unacceptable rates, although the average ranges from 22.3 % to 35.2 %. This dependence for the standard DBSCAN (Table III) is much worse at its average. The DBSCAN* averaged inaccuracy is also unacceptable at any cluster size (Table IV).

TABLE II
1-TUNED DBSCAN INACCURACY VERSUS CLUSTER SIZE

Points in cluster, thousands	Minimum	Average	Maximum	Standard deviation
1	0	35.1524	89.4556	27.567
2	0	30.0864	88.2389	27.8728
3	0	22.2566	94.4444	25.138
4	0	23.305	90.2875	26.1915
5	0	25.8529	87.1443	25.9151
6	0	22.6197	88.8889	25.09
7	0	30.4215	93.75	27.3401
8	0	26.8843	90.555	26.6312
9	0	25.5344	85.5457	26.081
10	0	29.3496	88.8906	26.7412

TABLE III
DBSCAN INACCURACY VERSUS CLUSTER SIZE

Points in cluster, thousands	Minimum	Average	Maximum	Standard deviation
1	0	62.6627	99.7235	35.5609
2	0	56.5287	99.405	39.5669
3	0	50.7411	99.3784	39.9859
4	0	48.0162	97.55	40.1831
5	0	52.0271	97.8586	39.8846
6	0	48.9661	94.5056	39.9975
7	0	58.2835	97.1657	37.7901
8	0	54.8974	94.4625	39.5227
9	0	52.0105	94.5531	39.6756
10	0	57.957	95.21	38.5499

TABLE IV
DBSCAN* INACCURACY VERSUS CLUSTER SIZE

Points in cluster, thousands	Minimum	Average	Maximum	Standard deviation
1	0	51.354	94.1176	35.1789
2	0	42.435	94.3472	37.4062
3	0	33.4541	94.4444	36.21
4	0	29.8574	94.3833	36.2104
5	0	34.9762	94.4444	36.7743
6	0	31.2271	94.1176	36.4199
7	0	38.5605	94.4444	37.4931
8	0	35.4963	94.1176	37.1912
9	0	33.4505	94.4444	36.7577
10	0	37.9878	94.4444	37.8465

The standard deviation value in Table I seemingly correlates with the average: the lower deviation value is typical for the lower average. In Tables II–IV, the standard deviation does not correlate with the average. The lowest standard deviation is at the 1-tuned DBSCAN.

The 1-tuned DBSCAN averaged inaccuracy ρ^*_{miss} presented in Table V grows as the number of clusters increases. The inaccuracy average increases from 15.1 % to 38.5 %. The similar trend is observable for the standard DBSCAN (Table VI) whose inaccuracy average increases from 41 % to 64.2 %. The DBSCAN* averaged inaccuracy increases from 21.1 % to 50.1 % (Table VII). The standard deviation in Tables V–VII increases as the number of clusters increases.

TABLE V
1-TUNED DBSCAN INACCURACY VERSUS NUMBER OF CLUSTERS

Number of clusters	Minimum	Average	Maximum	Standard deviation
3	0	15.0685	66.6667	16.0574
4	0	16.8318	61.775	17.3966
5	0	20.3785	80	20.5663
6	0	20.9759	83.3333	22.4075
7	0	22.7907	85.7143	23.8897
8	0	22.8558	78.765	24.0487
9	0	25.611	88.8889	25.7638
10	0	26.6221	90.2875	25.6458
11	0	27.7483	87.2736	26.4192
12	0	30.3219	90.1438	28.2461
13	0	30.6833	84.6659	27.8859
14	0	31.8776	87.1443	28.7255
15	0	33.6377	90.555	30.1265
16	0	33.9949	93.75	29.9216
17	0	36.4437	91.1782	30.4509
18	0	38.4988	94.4444	30.7374

TABLE VI
DBSCAN INACCURACY VERSUS NUMBER OF CLUSTERS

Number of clusters	Minimum	Average	Maximum	Standard deviation
3	0	40.9907	90.6667	31.3026
4	0	45.1063	93.375	33.9527
5	0	48.5407	97.1657	35.5419
6	0	46.3278	97.75	38.1635
7	0	50.1376	94.65	38.4128
8	0	49.4607	97.075	40.1362
9	0	52.8605	98.5	40.0237
10	0	54.6229	99.405	40.1863
11	0	57.7633	98.6273	38.4918
12	0	58.9166	98	39.5244
13	0	56.7122	98.2385	40.615
14	0	57.7996	98.8714	40.678
15	0	59.4278	98.2533	40.5949
16	0	61.2276	99.7	41.0152
17	0	63.2754	99.7235	40.7469
18	0	64.175	99.1083	39.6105

TABLE VII
DBSCAN* INACCURACY VERSUS NUMBER OF CLUSTERS

Number of clusters	Minimum	Average	Maximum	Standard deviation
3	0	21.1497	66.6667	26.3408
4	0	24.5782	75	30.4204
5	0	27.8073	80	31.7332
6	0	27.1963	83.3333	32.7129
7	0	32.6474	85.7143	35.0519
8	0	32.2856	87.5	35.7743
9	0	35.7875	88.8889	37.0391
10	0	37.8782	90	36.191
11	0	39.5002	90.9091	37.3249
12	0	40.4594	91.6667	38.7459
13	0	41.9702	92.3077	38.4366
14	0	42.5297	92.8571	38.645
15	0	44.3486	93.3333	39.3817
16	0	44.6521	93.75	40.9259
17	0	47.1443	94.1176	39.1954
18	0	50.1439	94.4444	39.7384

The percentage of missed points expectedly grows as the noise magnitude increases. The 1-tuned DBSCAN averaged inaccuracy increases from 16.3 % to 48.1 % (Table VIII), and its standard deviation insignificantly varies between 22.1 and 26.4. The standard DBSCAN averaged inaccuracy increases from 35.9 % to 81 % (Table IX), and its standard deviation varies in a much wider range from about 16.7 up to 41.2. The DBSCAN* averaged inaccuracy increases from 20.8 % to 68 % (Table X), although its standard deviation varying from 25.2 up to 36.8 has a wider range compared to the standard deviation in Table VIII of the 1-tuned DBSCAN. The 1-tuned DBSCAN has the lowest standard deviation whose range is the narrowest. The widest range is at the standard deviation of the standard DBSCAN.

TABLE VIII
1-TUNED DBSCAN INACCURACY VERSUS NOISE MAGNITUDE

Noise coefficient γ_2	Minimum	Average	Maximum	Standard deviation
0.1	0	16.3215	80	22.0532
0.2	0	17.8131	83.0961	23.2671
0.3	0	22.7937	89.4556	25.2922
0.4	0	30.7512	93.75	26.3728
0.5	0.04	48.052	94.4444	23.3163

TABLE IX
DBSCAN INACCURACY VERSUS NOISE MAGNITUDE

Noise coefficient γ_2	Minimum	Average	Maximum	Standard deviation
0.1	0	35.9387	98.25	40.8621
0.2	0	38.8354	99.3784	41.1857
0.3	0	48.5869	99.3396	40.2098
0.4	0	66.6652	99.7235	31.6883
0.5	0.03	81.019	99.7	16.6551

TABLE X
DBSCAN* INACCURACY VERSUS NOISE MAGNITUDE

Noise coefficient γ_2	Minimum	Average	Maximum	Standard deviation
0.1	0	20.8147	94.4444	32.2438
0.2	0	23.7643	94.2389	34.1033
0.3	0	30.0269	94.4444	35.6364
0.4	0	41.7464	94.4444	36.7843
0.5	0.03	68.0472	94.4444	25.1915

Thus, Tables I–X confirm that the 1-tuned DBSCAN is more accurate than DBSCAN*, i.e., it is more efficient to apply a tuned DBSCAN to multiple sub-datasets than tuning the standard DBSCAN over the whole dataset with subsequent application. Besides, compared to the DBSCAN* approach, datasets are clustered by the 1-tuned DBSCAN extremely fast. Therefore, Table XI shows that the ratio of DBSCAN* average runtime to 1-tuned DBSCAN average runtime increases from 14 to almost 65 as the cluster size increases. This ratio increases from 16.5 to 58.3 as the number of clusters increases (Table XII). There is no distinct trend in this ratio versus the noise magnitude (Table XIII). The standard deviation value in Tables XI and XII distinctly grows as the cluster size increases and the number of clusters increases.

TABLE XI
RATIO OF DBSCAN* TO 1-TUNED DBSCAN RUNTIME VERSUS CLUSTER SIZE

Points in cluster, thousands	Minimum	Average	Maximum	Standard deviation
1	5.0365	14.1306	18.6867	1.9133
2	10.2991	16.7943	21.7279	2.2837
3	12.7777	19.4851	26.7843	3.2439
4	13.3271	22.001	46.2373	4.494
5	14.5227	25.0742	42.4828	5.5725
6	14.2891	28.5536	61.9834	7.6331
7	12.9579	31.5921	54.5336	8.8613
8	16.6587	40.9372	176.6998	37.4835
9	14.157	53.7414	186.6482	51.8181
10	15.3662	64.821	724.7633	74.8311

The averaged best value of the 1-tuned DBSCAN neighbourhood radius is shown in Table XIV versus the noise magnitude. The average of ε^* ranges from 0.9677 to 1.6979 having a trend to decrease as the noise magnitude increases. Overall, ε^* ranges from 0.41 to 3.06. The best value of the DBSCAN* neighbourhood radius shown in Table XV versus the noise magnitude ranges farther up to 3.46. The range of the average of ε^* here is narrower – it ranges from 1.1693 to 1.4249 having the same trend to decrease. No trends for the standard deviation in Tables XIV and XV are noticed.

TABLE XII

RATIO OF DBSCAN* TO 1-TUNED DBSCAN RUNTIME VERSUS NUMBER OF CLUSTERS

Number of clusters	Minimum	Average	Maximum	Standard deviation
3	5.0365	16.5022	27.8287	3.1114
4	9.1092	17.7431	31.7187	3.448
5	10.0295	19.1869	38.8355	4.2098
6	9.8397	20.4585	35.688	5.026
7	10.4117	21.6811	43.5492	5.7892
8	10.9813	22.772	45.6321	6.1019
9	11.4521	24.1676	61.9834	6.9686
10	11.212	25.5865	56.5879	7.6422
11	12.3184	27.1494	109.839	10.018
12	11.0075	28.1425	48.7161	9.3782
13	13.2307	30.1024	56.9399	10.888
14	7.7181	41.1725	724.7633	55.9985
15	13.6071	49.6267	719.7899	63.6204
16	13.738	48.4227	189.2232	48.6839
17	13.9794	56.4355	190.9218	57.0504
18	14.7466	58.2593	195.5119	58.6954

TABLE XIII

RATIO OF DBSCAN* TO 1-TUNED DBSCAN RUNTIME VERSUS NOISE MAGNITUDE

Noise coefficient γ_2	Minimum	Average		Maximum	Standard deviation
0.1	5.0365	32.8327		719.7899	40.1755
0.2	9.2544	31.4171		192.9775	31.6045
0.3	10.393	31.242		191.6876	31.425
0.4	8.8297	31.2208		192.2546	31.6505
0.5	7.7181	31.8527		724.7633	39.9081

TABLE XIV

BEST VALUE OF 1-TUNED DBSCAN NEIGHBOURHOOD RADIUS

Noise coefficient γ_2	Minimum	Average	Maximum	Standard deviation
0.1	0.66	1.6979	2.91	0.3332
0.2	0.51	1.5412	2.41	0.3178
0.3	0.41	1.3386	2.36	0.3104
0.4	0.41	1.1376	2.91	0.3059
0.5	0.51	0.9677	3.06	0.3512
Overall	0.41	1.3366	3.06	0.4179

TABLE XV

BEST VALUE OF DBSCAN* NEIGHBOURHOOD RADIUS

Noise coefficient γ_2	Minimum	Average	Maximum	Standard deviation
0.1	0.41	1.4249	3.66	0.5698
0.2	0.41	1.3322	3.71	0.5186
0.3	0.41	1.2157	3.71	0.5068
0.4	0.41	1.1693	3.36	0.6529
0.5	0.51	1.3171	3.46	0.9062
Overall	0.41	1.2919	3.71	0.654

The instance clustered the worst by the 1-tuned DBSCAN is shown in Fig. 4, which resembles Fig. 2 by those tightly packed serpentine clusters. Although the number of outliers is much fewer (here $\rho^*_{\text{out}} = 2.6229$), nearly a half of the points are assigned to non-existing clusters and more than a third are confused:

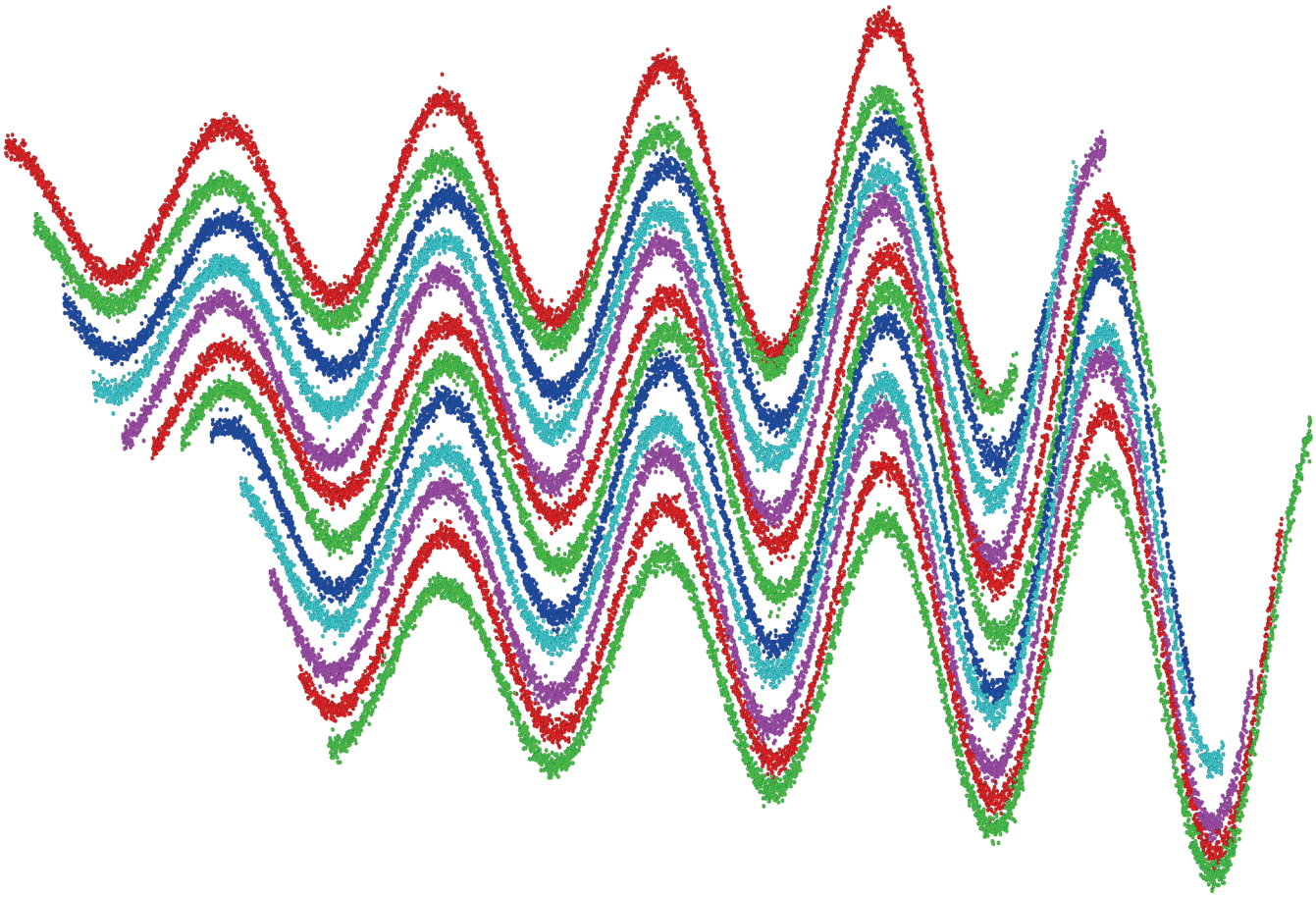

Fig. 4. The serpentine dataset of 48 000 points within 12 clusters, 4000 points in each cluster, which is clustered by the 1-tuned DBSCAN with the worst inaccuracy. Generated at the noise maximum, the dataset has a few regions with tightly packed clusters resulting in 90.1437 % of points are missed.

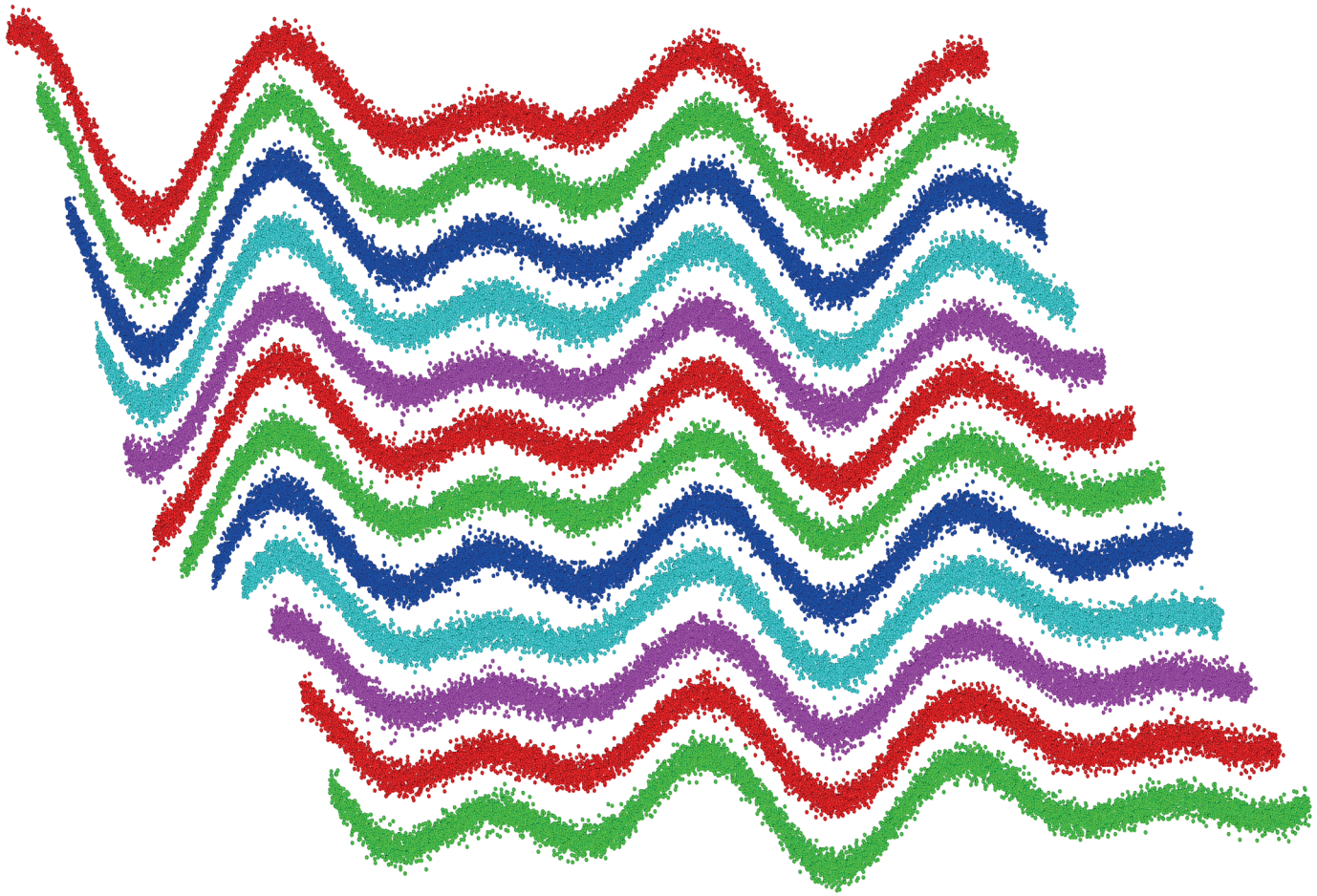

Fig. 5. The serpentine dataset of 108 000 points within 12 clusters, 9000 points in each cluster, is clustered by the 1-tuned DBSCAN at an acceptable inaccuracy. DBSCAN* fails to cluster the dataset properly – having no assignments to non-existing clusters, it misses 41.8 % of the dataset points.

$$\rho^*_{oth} = 50.8083\,,\ \rho^*_{conf} = 36.7125\,,\ \rho^*_{miss} = 90.1437\,,$$

where $\varepsilon^* = 0.61$. DBSCAN* is even more inaccurate – having no outliers and assignments to non-existing clusters, it confuses 91.6667 % of points at $\varepsilon^* = 3.26$. Nevertheless, even at the noise maximum some instances are clustered at an acceptable inaccuracy. Thus, Fig. 5 shows a dataset of 108 000 points whose 12 clusters are determined sufficiently accurately at $\varepsilon^* = 0.71$:

$$\rho^*_{out} = 0.3065\,,\ \rho^*_{oth} = 1.6602\,,\ \rho^*_{conf} = 4.1574\,,\ \rho^*_{miss} = 6.1241\,.$$

Despite having no assignments to non-existing clusters, DBSCAN* is far more inaccurate:

$$\rho^*_{out} = 0.2352\,,\ \rho^*_{oth} = 0\,,\ \rho^*_{conf} = 41.5648\,,\ \rho^*_{miss} = 41.8\,.$$

Similarly generated worst-case scenarios of the clustering algorithm performance confirm the 1-tuned DBSCAN supremacy, although fails like that in Fig. 4 are not excluded.

V. Discussion

The series of computational simulations shows that the DBSCAN algorithm is sped up for clustering time-serpentine datasets by using the best neighbourhood radius and applying the so-tuned DBSCAN to a set of sub-datasets. The factual speedup registered (Tables XI–XIII) ranges from 5.0365 to 724.7633 having a trend to increase as the dataset size increases. The DBSCAN algorithm updated in the way of dataset division and preliminarily determining the best value of the neighbourhood radius gains in accuracy as well.

As of this research, the 1-tuned DBSCAN application seems to be limited to tightly packed time-serpentine datasets of large size. Such datasets typically emerge in estimating computer vision-based position for autonomous vehicles, adaptive traffic pattern recognition, tracking applications, etc. [12], [28], [30], [42]. Another limitation is that the minimum and maximum values in the neighbourhood radius range depend on the dataset scale, so they must be selected manually. The same relates to the step by which the neighbourhood radius is incremented while problem (2) is solved numerically [43], [44]. As this step is set to a smaller value, the speedup will be lower, but the accuracy gain will be higher. Step $\Delta\varepsilon = 0.1$ is fairly rough for the serpentine datasets generated by (9), (11), (8), (10), (12)–(17) results. Setting $\Delta\varepsilon = 0.01$ improves most of cluster results of the 80 000 instances, although the 1-tuned DBSCAN runtime elongates (not by 10 times) because determining the best neighbourhood radius is done over a 10-times denser subset of the interval between ε_{min} and ε_{max}.

VI. Conclusion

In order to partition a tightly packed time-serpentine dataset of large size, the dataset is initially divided into a few sub-datasets along the time axis, whereupon the best neighbourhood radius is determined over the first sub-dataset and the standard DBSCAN algorithm is run over all the sub-datasets by the best neighbourhood radius. A reasonable trade-off for the neighbourhood radius step and the DBSCAN speedup must be found. The method of determining the best neighbourhood radius, at which DBSCAN misses points of the first sub-dataset the least, can be any numerical approach to finding the global minimum of a tabulated function.

Although it is necessary to know ground truth cluster labels of points within a region, the first sub-dataset is not necessary to be the one. However, knowing ground truth cluster labels of points at dataset boundaries is important to tune the neighbourhood radius appropriately.

Specifically for tightly packed time-serpentine large datasets, the suggested approach contributes to a faster DBSCAN clustering in applied computer science. Its practical application lies in more efficient performance of tracking objects on a two-dimensional map whose traces highly correlate. Further research must be built on studying possibility to tune the neighbourhood radius for non-correlated clusters. Beyond this, the case of three and more features should be studied.

References

[1] 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 Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, AAAI Press, 1996, pp. 226–231. [Online]. Available: https://file.biolab.si/papers/1996-DBSCAN-KDD.pdf

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

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

[4] 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*, vol. 28, no. 1, pp. 1–12, Jun. 2023. https://doi.org/10.2478/acss-2023-0001

[5] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "DBSCAN Revisited, revisited: Why and how you should (still) use DBSCAN," *ACM Transactions on Database Systems*, vol. 42, no. 3, Jul. 2017, Art. no. 19. https://doi.org/10.1145/3068335

[6] N. Hanafi and H. Saadatfar, "A fast DBSCAN algorithm for big data based on efficient density calculation," *Expert Systems with Applications*, vol. 203, Oct. 2022, Art. no. 117501. https://doi.org/10.1016/j.eswa.2022.117501

[7] 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*, vol. 2, no. 2, pp. 169–194, Jun. 1998. https://doi.org/10.1023/A:1009745219419

[8] X. Huang, T. Ma, C. Liu, and S. Liu, "GriT-DBSCAN: A spatial clustering algorithm for very large databases," *Pattern Recognition*, vol. 142, Oct. 2023, Art. no. 109658. https://doi.org/10.1016/j.patcog.2023.109658

[9] S. Pourbahrami, "A neighborhood-based robust clustering algorithm using Apollonius function kernel," *Expert Systems with Applications*, vol. 248, Aug. 2024, Art. no. 123407. https://doi.org/10.1016/j.eswa.2024.123407

[10] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A *k*-means clustering algorithm," *Journal of the Royal Statistical Society, Series C*, vol. 28, no. 1, pp. 100–108, 1979. https://doi.org/10.2307/2346830

[11] A. M. Ikotun, A. E. Ezugwu, L. Abualigah, B. Abuhaija, and J. Heming, "*K*-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data," *Information Sciences*, vol. 622, pp. 178–210, Apr. 2023. https://doi.org/10.1016/j.ins.2022.11.139

[12] Z. Wei, Y. Gao, X. Zhang, X. Li, and Z. Han, "Adaptive marine traffic behaviour pattern recognition based on multidimensional dynamic time warping and DBSCAN algorithm," *Expert Systems with Applications*, vol. 238, Part E, Mar. 2024, Art. no. 122229. https://doi.org/10.1016/j.eswa.2023.122229

[13] J. Xie, L. Jiang, S. Xia, X. Xiang, and G. Wang, "An adaptive density clustering approach with multi-granularity fusion," *Information Fusion*, vol. 106, 2024, Jun. Art. no. 102273. https://doi.org/10.1016/j.inffus.2024.102273

[14] B. Ma, C. Yang, A. Li, Y. Chi, and L. Chen, "A faster DBSCAN algorithm based on self-adaptive determination of parameters," *Procedia Computer Science*, vol. 221, pp. 113–120, 2023. https://doi.org/10.1016/j.procs.2023.07.017

[15] Y. Chen, L. Zhou, N. Bouguila, C. Wang, Y. Chen, and J. Du, "BLOCK-DBSCAN: Fast clustering for large scale data," *Pattern Recognition*, vol. 109, Jan. 2021, Art. no. 107624. https://doi.org/10.1016/j.patcog.2020.107624

[16] F. Ros, S. Guillaume, R. Riad, and M. El Hajji, "Detection of natural clusters via S-DBSCAN a self-tuning version of DBSCAN," Knowledge-Based Systems, vol. 241, Apr. 2022, Art. no. 108288. https://doi.org/10.1016/j.knosys.2022.108288

[17] D. Luchi, A. L. Rodrigues, and F. M. Varejão, "Sampling approaches for applying DBSCAN to large datasets," *Pattern Recognition Letters*, vol. 117, pp. 90–96, 2019. https://doi.org/10.1016/j.patrec.2018.12.010

[18] N. Newaliya and Y. Singh, "Multivariate hierarchical DBSCAN model for enhanced maritime data analytics," *Data & Knowledge Engineering*, vol. 150, Mar. 2024, Art. no. 102282. https://doi.org/10.1016/j.datak.2024.102282

[19] P. Sadhukhan, L. Halder, and S. Palit, "Approximate DBSCAN on obfuscated data," *Journal of Information Security and Applications*, vol. 80, Feb. 2024, Art. no. 103664. https://doi.org/10.1016/j.jisa.2023.103664

[20] J. Gan and Y. Tao, "DBSCAN revisited: Mis-claim, un-fixability, and approximation," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, May 2015, pp. 519–530. https://doi.org/10.1145/2723372.2737792

[21] T. Boonchoo, X. Ao, Y. Liu, W. Zhao, F. Zhuang, and Q. He, "Grid-based DBSCAN: Indexing and inference," *Pattern Recognition*, vol. 90, pp. 271–284, Jun. 2019. https://doi.org/10.1016/j.patcog.2019.01.034

[22] N. Gholizadeh, H. Saadatfar, and N. Hanafi, "K-DBSCAN: An improved DBSCAN algorithm for big data," *Journal of Supercomputing*, vol. 77, no. 6, pp. 6214–6235, June 2021. https://doi.org/10.1007/s11227-020-03524-3

[23] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: Ordering points to identify the clustering structure," in *ACM SIGMOD International Conference on Management of Data*, ACM Press, Jun. 1999, pp. 49–60. https://doi.org/10.1145/304181.304187

[24] R. J. G. B. Campello, D. Moulavi, A. Zimek, and J. Sander, "A framework for semi-supervised and unsupervised optimal extraction of clusters from hierarchies," *Data Mining and Knowledge Discovery*, vol. 27, no. 3, pp. 344–371, Apr. 2013. https://doi.org/10.1007/s10618-013-0311-4

[25] M. Al Samara, I. Bennis, A. Abouaissa, and P. Lorenz, "Complete outlier detection and classification framework for WSNs based on OPTICS," *Journal of Network and Computer Applications*, vol. 211, Feb. 2023, Art. no. 103563. https://doi.org/10.1016/j.jnca.2022.103563

[26] J. Wang, Z. Liu, Y. Zhao, Y. Xie, and Y. Xie, "EAST-NBI experimental data processing method based on improved OPTICS algorithm," *Fusion Engineering and Design*, vol. 172, Nov. 2021, Art. no. 112737. https://doi.org/10.1016/j.fusengdes.2021.112737

[27] V. V. Romanuke, "Uniform rectangular array radar optimization for efficient and accurate estimation of target parameters," *Information and Telecommunication Sciences*, vol. 13, no. 1, pp. 44–55, 2022. [Online]. Available: http://infotelesc.kpi.ua/article/view/259751/256220

[28] X. Bai, Z. Xie, X. Xu, and Y. Xiao, "An adaptive threshold fast DBSCAN algorithm with preserved trajectory feature points for vessel trajectory clustering," *Ocean Engineering*, vol. 280, Jul. 2023, Art. no. 114930. https://doi.org/10.1016/j.oceaneng.2023.114930
[29] V. V. Romanuke, "A prototype model for semantic segmentation of curvilinear meandering regions by deconvolutional neural networks," *Applied Computer Systems*, vol. 25, no. 1, pp. 62–69, May 2020. https://doi.org/10.2478/acss-2020-0008
[30] B. Żak and S. Hożyń, "Local image features matching for realtime seabed tracking applications," *Journal of Marine Engineering & Technology*, vol. 16, no. 4, pp. 273–282, Oct. 2017. https://doi.org/10.1080/20464177.2017.1386266
[31] V. V. Romanuke, "Accurate detection of multiple targets by uniform rectangular array radar with threshold soft update and area rescanning," *Information and Telecommunication Sciences*, vol. 13, no. 2, pp. 62–71, Dec. 2022. https://doi.org/10.20535/2411-2976.22022.62-71
[32] 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*, vol. 1, no. 6, pp. 263–265, 2018.
[33] S. J. Phillips, "Acceleration of *K*-Means and related clustering algorithms," in D. M. Mount and C. Stein, Eds., *Lecture Notes in Computer Science*, vol. 2409, Springer, Jan. 2002, pp. 166–177. https://doi.org/10.1007/3-540-45643-0_13
[34] 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*, vol. 28, no. 2, pp. 189–202, Dec. 2023. https://doi.org/10.2478/acss-2023-0019
[35] C. L. Valenzuela and A. J. Jones, "Evolutionary divide and conquer (I): A novel genetic approach to the TSP," *Evolutionary Computation*, vol. 1, no. 4, pp. 313–333, Dec. 1993. https://doi.org/10.1162/evco.1993.1.4.313
[36] V. V. Romanuke, "Traveling salesman problem parallelization by solving clustered subproblems," *Foundations of Computing and Decision Sciences*, vol. 48, no. 4, pp. 453–481, Dec. 2023. https://doi.org/10.2478/fcds-2023-0020
[37] V. V. Romanuke, "Deep clustering of the traveling salesman problem to parallelize its solution," *Computers & Operations Research*, vol. 165, May 2024, Art. no. 106548. https://doi.org/10.1016/j.cor.2024.106548
[38] T. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical Computer Science*, vol. 38, pp. 293–306, 1985. https://doi.org/10.1016/0304-3975(85)90224-5
[39] A. Czapiewska, A. Luksza, R. Studanski, and A. Żak, "Reduction of the multipath propagation effect in a hydroacoustic channel using filtration in cepstrum," *Sensors*, vol. 20, iss. 3, Jan. 2020, Art. no. 751. https://doi.org/10.3390/s20030751
[40] Y. Zack, "Cluster analysis for multidimensional objects in fuzzy data conditions," *System research and information technologies*, no. 2, pp. 18–34, Dec. 2021. https://doi.org/10.20535/SRIT.2308-8893.2021.2.02
[41] G. Grzeczka and M. Klebba, "Automated calibration system for digital multimeters not equipped with a communication interface," *Sensors*, 2020, vol. 20, iss. 13, Jun. 2020, Art. no. 3650. https://doi.org/10.3390/s20133650
[42] J. Zalewski and S. Hożyń, "Computer vision-based position estimation for an autonomous underwater vehicle," *Remote Sensing*, vol. 16, iss. 5, Feb. 2024, Art. no. 741. https://doi.org/10.3390/rs16050741
[43] W. Kaplan, "Maxima and minima with applications: Practical optimization and duality," in *Wiley Series in Discrete Mathematics and Optimization*, vol. 51, John Wiley & Sons, 2011, p. 61.
[44] V. V. Romanuke, "Three-point iterated interval half-cutting for finding all local minima of unknown single-variable function," *Electrical, Control and Communication Engineering*, vol. 18, no. 1, pp. 27–36, Jun. 2022. https://doi.org/10.2478/ecce-2022-0004

Vadim V. Romanuke was born in 1979. He graduated from the Technological University of Podillya in 2001. In 2006, he received the Degree of Candidate of Technical Sciences in Mathematical Modelling and Computational Methods. The Candidate Dissertation suggested a way of increasing interference noise immunity of data transferred over radio systems. The degree of Doctor of Technical Sciences in Mathematical Modelling and Computational Methods was received in 2014. The Doctor-of-Science Dissertation solved a problem of increasing efficiency of identification of models for multistage technical control and run-in under multivariate uncertainties of their parameters and relationships. In 2016, he received the status of a Full Professor.
He is a Professor of the Faculty of Mechanical and Electrical Engineering at the Polish Naval Academy. His current research interests concern decision making, game theory, semantic image segmentation, statistical approximation, and control engineering based on statistical correspondence. He has 425 published scientific articles, one monograph, one tutorial, and 16 methodical guidelines in Functional Analysis, Mathematical and Computer Modelling Master Thesis development, Conflict-Controlled Systems, Algorithms and Data Structures, Data Analysis, Master Academic Practice. Until January 2018, Vadim Romanuke was the scientific supervisor of a Ukrainian budget grant work concerning minimization of water heat transfer and consumption.
Address for correspondence: 69 Śmidowicza Street, Gdynia, Poland, 81-127.
E-mail: romanukevadimv@gmail.com
ORCID iD: https://orcid.org/0000-0003-3543-3087