Data with Density-Based Clusters: A Generator for Systematic Evaluation of Clustering Algorithms

Philipp Jahn[1,2(✉)], Christian M. M. Frey[3], Anna Beer[4], Collin Leiber[1,2], and Thomas Seidl[1,2,3]

[1] LMU Munich, Munich, Germany
{jahn,leiber,seidl}@dbs.ifi.lmu.de
[2] Munich Center for Machine Learning (MCML), Munich, Germany
[3] Fraunhofer IIS, Erlangen, Germany
[4] University of Vienna, Vienna, Austria
anna.beer@univie.ac.at

Abstract. Mining data containing density-based clusters is well-established and widespread but faces problems when it comes to systematic and reproducible comparison and evaluation. Although the success of clustering methods hinges on data quality and availability, reproducibly generating suitable data for this setting is not easy, leading to mostly low-dimensional toy datasets being used. To resolve this issue, we propose DENSIRED (**DENSI**ty-based **R**eproducible **E**xperimental **D**ata), a novel data generator for data containing density-based clusters. It is highly flexible w.r.t. a large variety of properties of the data and produces reproducible datasets in a two-step approach. First, skeletons of the clusters are constructed following a random walk. In the second step, these skeletons are enriched with data samples. DENSIRED enables the systematic generation of data for a robust and reliable analysis of methods aimed toward examining data containing density-connected clusters. In extensive experiments, we analyze the impact of user-defined properties on the generated datasets and the intrinsic dimensionalities of synthesized clusters. Our code and novel benchmark datasets are publicly available at: https://github.com/PhilJahn/DENSIRED.

Keywords: Clustering · Evaluation · Data Generator · Density-based

1 Introduction

In order to properly develop and evaluate new clustering methods, access to suitable data is essential. However, real-world data is often limited in terms of availability, expensive to annotate, or even biased in certain ways [1]. By generating synthetic data, machine learning engineers and researchers can evaluate

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-70368-3_1.

A. Bifet et al. (Eds.): ECML PKDD 2024, LNAI 14947, pp. 3–21, 2024.
https://doi.org/10.1007/978-3-031-70368-3_1

their novel methods more thoroughly and in a controlled, reproducible, and flexible environment. The manual creation of such data is not easy for use cases like density-based clustering, where no suitable data generators exist. Existing generators are insufficient w.r.t. generating arbitrarily shaped clusters, guarantees of the cluster's density-separability, or the capability to generate similar, yet distinct datasets. State-of-the-art data generators for synthetic data are limited in terms of complexity, especially in higher-dimensional spaces: e.g., they cannot produce non-convex, high-dimensional clusters. Hence, we develop DENSIRED (**DENSI**ty-based **R**eproducible **E**xperimental **D**ata), a tool to synthesize data containing density-connected structures in high-dimensional spaces in a controlled, reproducible, and customizable way. Users can adjust a large set of potentially relevant properties of a dataset, e.g., its size, dimensionality, number of clusters, density of clusters, or noise ratio. DENSIRED can create density-connected clusters as known from DBSCAN [2] as well as other non-convex clusters of arbitrary shape, varying density, non-linearly separable clusters, and clusters with several branches. This set of properties is of special interest for evaluating advantages of density-based clustering methods. Using a two-step approach, DENSIRED separates the generation of the underlying structures ("skeletons") of clusters and the instantiation of the actual data points. This allows the synthetization of various datasets with the same overall structure that are equivalently hard to cluster. We define and show the importance of this property. The process is depicted in Fig. 1 along with some examples for the two steps.

We showcase various benefits of our data generator by performing a comprehensive and broad benchmarking study. In this study, we compare 14 algorithms based on diverse heuristics for clustering on various datasets. As DENSIRED is able to maintain the general structure with increasing dimensionality, interesting conclusions can be drawn about the behavior of the considered methods. The generated data exhibits the intended characteristics: density-based algorithms deliver substantially better results than, e.g., centroid-based methods. Our contributions are as follows:

- We introduce the novel data generator DENSIRED for synthesizing data that consists of density-connected structures in high-dimensional spaces
- DENSIRED supports the reproducible generation of clusters of arbitrary shapes, varying intrinsic dimensionality, different densities, and diverse geometric characteristics
- We perform a broad benchmark study on exemplary generated datasets to demonstrate the evaluation possibilities for novel clustering approaches

2 Related Work

Benchmark Data
There are numerous well-established benchmark datasets covering various domains for evaluating data mining methods.

1) Real-world data. Real-world datasets are, e.g., used for the evaluation of clustering approaches [2–5]. The inclusion of real-world datasets is essential for a

Table 1. Data properties in the evaluation of density-connectivity-based clustering methods (sorted by year of publication)

	synthetic		real
Method	2d	>2d	
DBSCAN [2]	✔	✗	✔
GDBSCAN [6]	✔	✗	✔
OPTICS [7]	✔	✗	✔
MDD [9]	✔	✗	✗
HDBSCAN [11]	✗	✗	✔
DBSCAN Revisited [14]	✔	✔	✔
AnyDBC [15]	✔	✔	✔
DSets-DBSCAN [16]	✔	✗	✔
RNN-DBSCAN [17]	✔	(✔)	✔
AA-DBSCAN [18]	✔	✔	✔
DDC [19]	✔	✗	✔
BLOCK-DBSCAN [20]	✔	✗	✔
HDBSCAN-MR [21]	✗	✔	✔
AMD-DBSCAN [22]	✔	✗	✔
GriT-DBSCAN [23]	✔	✔	✔
K-DBSCAN [24]	✔	✗	✔
MDBSCAN [25]	✔	✗	✔

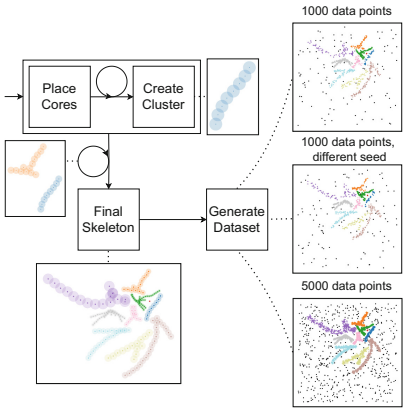

Fig. 1. Process of DENSIRED: Multiple clusters, each consisting of a set of cores, are created to produce a final skeleton. This can then be used to generate datasets of the same shape but with different data points (top and middle) and data numbers (bottom). The setting favors elongated datasets branching at the start point.

proper evaluation. Still, a systematic evaluation often requires specific tuning of data parameters, which can not easily be achieved with real-world datasets.

2) Synthetic data. To procure datasets of varying and sufficient complexity, synthetic datasets have been established as benchmarks. Widespread (but dated) datasets [37] allow examining methods' performances w.r.t. varying cardinality and number of clusters but are, for the most part, only 2-dimensional. Many are based on Gaussian distributions (including the few higher dimensional datasets), which do not necessarily conform with the density-based setting.

Evaluation of Density-Based Clustering

While many methods assume certain properties for density-based clusterings, they are often described only cursory, if at all. Often, such density-based clusters are described as sets of points that are separated from other sets of points by an area of low density. The most frequently used definition of density-based clusters describes them as a maximal set of density-connected points [2].

The concept of *density-connectivity* became famous with the introduction of DBSCAN [2] and builds upon the concept of *direct density-reachability*. Two points p and q are *directly density-reachable* iff they are both *core* points, i.e., they have at least $minPts$ many points within their ε-range. For density-connected points, it is sufficient if there exists a chain of directly density-reachable core points between them. There are plenty clustering methods that aim to detect density-connected clusters, e.g.: DBSCAN [2], GDBSCAN [6], Multi-density DBSCAN (MDD) [9].

Table 2. Properties of selected data generators for data with >2 dimensions

Name	Arbitrarily shaped underlying model?	Not linearly separable?	Guaranteed density separability?	Ability to generate equivalent datasets? (see Sect. 3)
Milligan's [26]	✗	✗	✔	✗
Improved Milligan's [27]	✗	✗	✔	✗
MixSim [28]	✗	✔	✗	✗
swiss-roll [29]	✗	✔	✗	✔
blobs [29]	✗	✔	✗	✔
Clugen [30]	✗	✔	✗	✔
OCLUS [31]	✗	✔	✔	✗
HAWKS [32]	✗	✔	✔	✗
MDCGEN [33]	✗	✔	✔	✗
SynDECA (Irregular 2) [34]	✔	✗	✔	✗
SynDECA (Irregular 1) [34]	✔	✔	✗	✗
Seed Spreader [14,35]	✔	✔	✗	✗
DataGen [36]	✔	✔	✗	✔
DENSIRED	✔	✔	✔	✔

Both, real-world and synthetic data, are used to evaluate these methods, as shown in Table 1. High-dimensional synthetic data is used least commonly and the way these datasets are obtained also varies, e.g., making use of specific benchmarking datasets [18,37], creating their own data [15], or using data generators [14,17,21,23,38]. Seed Spreader [14,35], as a generator for specifically density-based data, was used in [14,23]. RNN-DBSCAN [17] was evaluated using 3-dimensional data produced through scikit-learn[1] [29] (swiss-roll (3d) and 3d blobs). MixSim [28] was used for HDBSCAN-MR [21]. Still, many evaluations rely on 2d synthetic data for benchmarking. While evaluating on real-world data is essential, not all real-world datasets are equally suitable for density-connectivity-based clustering, as pointed out in [39]. Certain relevant properties, like high variability in density [22], cannot be easily evaluated in a systematic manner solely based on real-world data. As most properties of real-world data can not be tuned, their distinct effects on the quality of novel methods are hard to observe. Thus, there is a need for high-dimensional density-based synthetic data.

Data Generators

Data-generating functions have been established as a useful tool for a controlled evaluation of procedures in a plethora of domains ranging from healthcare [12],

[1] https://scikit-learn.org, last accessed: Oct 5th, 2023.

routing problems [10], flow networks [8], to manufacturing [13]. However, in contrast to data-driven generators (data simulators) that aim at recreating observed data, we regard process-driven data generators that are based on computational models [40]. This allows a generalized generative system that can produce a large variety of datasets containing density-based structures and is not limited to a specific use case. However, many existing generators cannot synthesize the required high-dimensional data (see Table 1). E.g., Pei's generator [41] offers arbitrarily shaped clusters but is restricted to two dimensions.

Table 2 shows which generators can create data with the denoted important properties for evaluating density-based clustering:

1) Arbitrarily shaped underlying model. In contrast to k-Means [42], density-based clustering methods are capable of dealing with arbitrarily shaped clusters [43,44]. To ensure that this ability is properly evaluated when investigating density-based methods, the generation of clusters with arbitrary shapes, e.g., by using random walks, is important. However, many generators [26–28,30–33] rely solely on using distributions with a singular midpoint for each component. Relying on a single distribution per dimension is not sufficient to generate arbitrary cluster shapes.

2) Not linearly separable. We want to generate data with clusters that are not pairwise linearly separable to enhance differences to centroid-based clustering methods. Clusters that are pairwise linearly separable can be detected easily by traditional, non-density-based clustering algorithms like k-Means [42]. The way to prevent overlap employed by some methods [26,27,34] causes them to always generate data that can be linearly separated.

3) Guaranteed density separability. Many algorithms do not have sufficient overlap handling [14,28,30] to guarantee that data from different clusters is not density-connected, leading to a flawed ground truth. The goal is to evaluate methods that use density-based assumptions. As a result, it leads to a lowered usability of the produced datasets and misleading results if these do not apply.

4) Ability to generate equivalent datasets. For a robust analysis, creating datasets that are equivalently hard to cluster is important, see Sect. 3. This requires either restating certain properties of the dataset [29] or storing and re-using the underlying model [30,36]. Some data generators, like MixSim [28], come close to fulfilling this requirement but cannot guarantee it. The property makes it possible to generate new datasets that share the same underlying structure while containing different points. This allows for, e.g., the expansion of existing datasets with additional points without generating the data points from new distributions or by excluding some data points initially and adding them again later on.

SynDECA [34] can generate "irregularly" [34] shaped clusters. However, they are either surrounded by hyperrectangular bounding boxes for overlap prevention (Irregular 2), making them easily separable, or potentially overlap (Irregular 1) due to violation of those bounding boxes. Datagen [36] fulfills desired properties but tends towards generating disconnected subclusters separated by other clusters. Seed Spreader [14,35] aims at creating high-dimensional synthetic data for

density-connectivity-based clustering methods. It is, thus, both our inspiration and main competitor. Like DENSIRED, Seed Spreader uses a random walk and generates its data points uniformly distributed within hyperspheres along this path. However, Seed Spreader generally offers very little control over the generation process and can lead to clusters that are not density-separable. We describe its deficits, which we overcome with DENSIRED, in detail in Sect. 3.4.

3 A Reliable Data Generator for Density-Based Clusters

For a robust analysis of clustering models, we want to examine datasets that are similar, but not equal. This alleviates effects of outliers or random events.

DENSIRED creates datasets containing density-connected clusters as known from, e.g., DBSCAN [2]. Note that this differs from mode-seeking density-based concepts as, e.g., used in Mean Shift [45]. The ε-neighborhood $N_\varepsilon(p)$ of a point p in a dataset X is $N_\varepsilon(p) = \{q \in X : dist(p,q) \leq \varepsilon\}$ for $\varepsilon > 0$ and a distance metric $dist(\cdot,\cdot)$. A point is a *core* point w.r.t. $\varepsilon \in \mathbb{R}^+$ and $\mu \in \mathbb{N}$ iff it has at least μ points in its ε-neighborhood. A point q is *directly density-reachable* from a point p for some $\varepsilon > 0$, $\mu \in \mathbb{N}$ iff p is in the ε-neighborhood of q and q is a core point. They are *density-reachable* iff there is a chain of directly density-connected core points between them and they are *density-connected* iff there is a point o such that p and q are both density-reachable from o. A (density-based) cluster $C^{\varepsilon,\mu} \subseteq X$ is a "maximal set of density-connected objects" [15].

For analyzing centroid-based clustering algorithms, clusters are usually generated by drawing from a random distribution. The resulting datasets are similar, but not equal. We define equivalence between two density-based clusterings:

Definition (Equivalence of density-based clusterings) Two datasets X, Y are equivalent regarding their density-connectivity if, for every ε and μ, there exist δ, ν, such that the ε, μ- clustering of X yields corresponding clusters to the δ, ν-clustering of Y. I.e., there is a bijective mapping between the density-connected components of both datasets.

For an unbiased evaluation of a novel method, it is important to eliminate unwanted random effects, like the single-link effect, and prevent overoptimism [46]. Thus, we need to carefully model the data in order to create equivalent datasets. For this reason, generating equivalent density-based clusters is a challenging problem because of its discrete, non-continuous aspect expressed by the parameter $\mu \in \mathbb{N}$.

3.1 Main Concept of DENSIRED

We construct equivalent but not equal density-connected clusters by following a two-step approach: 1) Defining the data distribution, i.e., the "skeleton" of the data, and 2) Instantiating the data points within this distribution. By separating the generation of the underlying dense areas from the actual instantiation of the data objects, we allow the synthesis of datasets with similar properties (e.g.,

shape and distance between clusters) but different instantiations. The goal is to produce arbitrarily shaped clusters that are density-connected even for higher dimensionalities. Preventing overlap is needed in order to ensure that the ground truth corresponds to the generated data, but the risk of the dataset being trivially separable due to the clusters being too far apart needs to be addressed.

As density-connected clusters do not follow any predefined shape or correlation, a random walk in high-dimensional space is a suitable model for generating the cluster's expansion [14]. In step one, we create a skeleton for each cluster with a random walk of a given step size that outputs core points. These are not necessarily the only core points of a cluster, but rather a subset that is sufficient to define a cluster's shape (cf. Section 3.2). In step two, we instantiate the data by filling the hypersphere of radius ε around each core point (cf. Section 3.3). This ensures that each core point has enough points in its ε-range. Each instantiation produces different datasets because of the randomness within each hypersphere. Clusters of different instantiations are equivalent and have the same shape in high-dimensional space but have deviations that are sufficient to avoid unwanted random effects. We offer users a large variety of optional parameters to customize data's properties, which are explained in the next subsections. A simplified pseudo code of DENSIRED is given in Algorithm 1, though lacks, e.g., restarting and noise adjustment.

3.2 Generation of Skeletons

In the first step, the local hyperspheres, also referred to as cores, are used as a skeleton to define the data distribution without sampling the actual data. The creation of the skeleton is based on a random walk assumption.

- **Number of clusters and cores** The number of clusters and cores can be defined by users. They can either specify the number of cores per cluster, or each cluster receives a random ratio of the overall core number. The starting positions of each cluster random walk can be randomly assigned or placed between two different clusters' random cores to increase the closeness of clusters. In case the second option fails because of, e.g., overlapping clusters, a random starting position is chosen for this cluster. If no position allows to start a new cluster, the value range ds in which a cluster can start is increased.
- **Momentum factor** ω Users can bias the random walk's direction of each step with a momentum factor $\omega \in [0, 1]$ individually per cluster. The default value is 0.5. Lower values lead to a rather randomly chosen direction for each step, while higher values keep the momentum and the direction $\boldsymbol{v}_{prev}$ of previous steps. For this, each step's direction is given by the weighted sum of a random vector $\boldsymbol{v}_{rand}$ and the previous step's direction vector, i.e., $\boldsymbol{v}_{new} = (1 - \omega)\boldsymbol{v}_{rand} + \omega\boldsymbol{v}_{prev}$.
- **Different Densities** DENSIRED allows to create clusters of varying density: scale factors for every cluster adjust the ε-radius of the cores and the distance between cores of the same cluster. Users can set these factors for every cluster separately, as a fixed value across all clusters, assign them randomly, or disable them to maintain the same density factor (default: 1) for all clusters.

- **Branching β and Stars $\varkappa$** The restart probability at every step of the random walk is determined by the branch factor β (default: 0.05). Restarts continue the random walk starting from an earlier step of the current walk. A star factor $\varkappa$ (default: 0) can be used to give the initial core of the walk a higher probability than other cores, else any of the cores is chosen randomly. Restarts are also performed whenever the random walk comes too close to another cluster's skeleton such that they keep a determined separability and do not overlap.
- **Overlapping Clusters** The separability between two clusters can be determined by a factor o (default: 1.1). Values less than 1 can lead to overlapping core spheres of different clusters. If a user-given amount of random walk step attempts failed due to coming too close to a core of another cluster, the random walk continues from a different core. If this fails, the cluster is fully restarted.

While the previous steps ensure the creation of clusters that fulfill the notion of density-based clusters, DENSIRED also allows to include properties that make the clustering task more challenging:

- **Step Sizes** The step size δ between consecutive cores can be set by users. Additionally, DENSIRED offers a flag to vary it slightly by choosing values from a truncated Gaussian distribution around the original step size δ^o, where $\frac{2}{3} \cdot \delta^o \leq \delta \leq 1.5 \cdot \delta^o$. Per default, this deviation is turned off.
- **Adding Chains.** DENSIRED can add linkages between clusters in the form of low-density paths between random points from different clusters. To create this path, the direction of each step of the random walk is randomly chosen, but the chain-specific version of the momentum factor ω is used to maintain the direction between the two clusters. To prevent a deadlock (due to, e.g., other clusters being in the way), any connection that fails too many times is dropped.

3.3 Instantiating Data Points

In the second step, data points within a skeleton's hypersphere are instantiated.

Users can define the approximate data's distribution among different clusters and noise. For each cluster, the number of points per core is computed based on this distribution and the size of the full dataset. For each core, the points are generated uniformly at random within the cluster-specific radius. Points at the center of the hypersphere to ensure density-connectivity for a given ε and $minPts$ are guaranteed by setting the flag *center*. Using the flag *equal* can ensure the same number of points per core of a cluster. Similar to the notion of [2], we define noise as additional random data points that are not density-connected to any cluster. We sample noise uniformly within the maximal bounds in high-dimensional space that is spanned by the radii of the hyperspheres of the cores. Additionally, noise points are not within a user-defined, cluster-dependent range of any cluster core.

Algorithm 1: Simplified Pseudo-Code for DENSIRED

1 *cores* ← {};
2 **for** *cluid* ∈ *range(cluster number)* **do**
3 Add core at starting position *pos* to *cores*[*cluid*];
4 $pos_{old} \leftarrow pos$; $\boldsymbol{v}_{old} \leftarrow \text{norm}(\boldsymbol{v}_{rand} - 0.5 \cdot \mathbf{1})$
5 **for** *coreid* ∈ *range(core number for cluid - 1)* **do**
6 $\boldsymbol{v} \leftarrow \text{norm}((1\text{-}\omega) \cdot \text{norm}(\boldsymbol{v}_{rand} - 0.5 \cdot \mathbf{1}) + \omega \cdot \boldsymbol{v}_{old})$; $pos \leftarrow pos_{old} + \delta \cdot \boldsymbol{v}$;
7 **if** *pos too close to other core in cores[¬cluid] based on* ε *and o, or due to* β **then**
8 Select pos_{old} based on $\varkappa$; $\boldsymbol{v} \leftarrow \text{norm}(\boldsymbol{v}_{rand} - 0.5 \cdot \mathbf{1})$; Go to 5;
9 **else**
10 add core at *pos* to *cores*[*cluid*]; $pos_{old} \leftarrow pos$; $\boldsymbol{v}_{old} \leftarrow \boldsymbol{v}$
11
12 **end for**
13 **end for**
14 Generate connections (3 to 12 with $\boldsymbol{v}_{old}$ pointing to a randomly chosen core of the target cluster);
15 Generate *data amount* · (1 - *noise ratio*) data points based on cluster cores in *cores* and ε of the cluster;
16 Generate *data amount* · *noise ratio* data points uniformly at random (while replacing those too close to cores);

3.4 Delimitations

Applying random walks with restarts and local hyperspheres within which data points are uniformly generated are based on ideas proposed for Seed Spreader [14]. However, there are crucial differences: Seed Spreader does not track individual hyperspheres and completely randomly expands the clusters, potentially leading to overlapping clusters, especially in lower dimensional space. This also means that Seed Spreader does not offer any control over the clusters' spatial expansion. Moreover, Seed Spreader only maintains a list of data points, limiting the ability to increase the number of data points generated from the same underlying data distributions, as further generation of data points will only randomly expand the latest existing clusters or add new clusters. The final number of clusters can not be set, but is instead based on a restart chance. Thus, any aspects on an individual cluster level can not be properly user-controlled. While density variance has been introduced in [35], the setting is limited to a fixed scaling factor on the size without affecting the number of points within a hypersphere.

We overcome these limitations as follows: DENSIRED follows a two-step approach, where the creation of the skeletons are decoupled from their instantiations. By tracking core points, DENSIRED controls the number of clusters and prevents cluster overlaps. Additional degrees of freedom are given by the parameters momentum ω, branching β, and the star factor $\varkappa$, which in combination impact the shape of (individual) clusters. Different cluster densities are

achieved through the relative ratio of data points and the size of hyperspheres. Lastly, the instantiation relies solely on the underlying distribution rather than changing a cluster's skeleton. This enables the generation of equivalent but not equal cluster structures.

3.5 Analysis Intrinsic Dimensionality

The intrinsic dimensionality (id) of a dataset [47] refers to the topological dimension of (nonlinear) manifolds a high-dimensional dataset can be mapped into. Given a dataset $X := \{x_i\}_{i=1}^{N} \subseteq \mathbb{R}^d$ in d-dimensional space, i.e., $dim(X) = d$, the aim of projective id estimators is to compute a mapping $\phi : x_i \to \tilde{x}_i \in \mathcal{M} \subseteq \mathbb{R}^l$ with $l \ll d$. The minimal number of vectors that linearly span the subspace $\mathcal{M}$ can be estimated by applying PCA. Assuming a data space $X \sim \mathcal{N}(\mu, \sigma)$ where samples are generated by a multivariate Gaussian distribution with covariance matrix $\Sigma \in \mathbb{R}^{p \times p}$ for which all pairs of components are uncorrelated and each component having the same variance. In the asymptotic case of $d \to \infty$, the empirical variances $\sigma_k, \forall k = 1, \ldots, p$, are converging to a unit value σ. In the case of zero correlation amongst the components the eigenvalues are equal to the respective standard deviation, i.e., the diagonal entries of the covariance matrix Σ. Moreover, the eigenvector matrix becomes an identity matrix leading to a permutation-invariant result, where each principal component contributes equally to the explained variance of X. Note that for a multivariate Gaussian distribution $X \sim \mathcal{N}(\mu, \sigma)$ with m components, and diagonal covariance matrix $\sigma \in \mathbb{R}^{p \times p}$ with variance $\sigma = \sigma_1 = \ldots = \sigma_m$, the $id(X) \propto dim(X)$.

Figure 2 shows the explained variance computed by a PCA w.r.t. the dimensions for a single cluster in $\mathbb{R}^{100}$ with 1000 cores. The datasets are scaled to a range between 0 and 1 based on their maximal values. The straight blue line corresponds to equally distributed principal components as described for the Gaussian distribution. *DENSIRED* shows the default setting of our approach, whereas for the other variants, all parameters are fixed besides the one indicated by the respective superscript. The impact of our generator's parameters on a cluster's id is as follows:

Increasing the momentum factor $\omega \to 1$ shown for $DENSIRED^{\omega}$ ($\omega = 1$, default: 0.5) diminishes the amount of uncertainty in the direction vector and, thus, emphasizes the most representative eigenvectors. The id(X) of a cluster X is reduced as a higher explained variance is already reached with a lower amount of dimensions being taken into account. Increasing the star factor $\varkappa \to 1$ for $DENSIRED^{\varkappa}$ ($\varkappa = 1$; default: 0) increases the probability that new core points are set in the ϵ-neighborhood of a cluster's initial core point. It results in star-like topologies where progressively, i.e., with a higher amount of the star factor, more eigenvectors need to be considered to explain the variance. Thus, it increases id(X) of a cluster X. Increasing the branching factor $\beta \to 1$ for $DENSIRED^{\beta}$ ($\beta = 1.0$; default: 0.05) increases the probability that new core points are set in the ε-neighborhood of a cluster's existing core points, where more eigenvectors are required to reach a higher amount of explained variance. It increases the id(X) of a cluster X. Lastly, increasing the step size δ in the set of core points

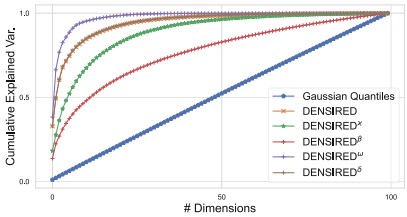

Fig. 2. Cum. explained var. on a single cluster in $\mathbb{R}^{100}$ showing the effects described in Sect. 3.5 (superscript refers to the regarded parameter)

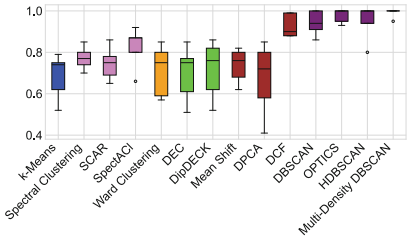

Fig. 3. NMI across all examined dimensionalities. Color indicates clustering concept as described in Sect. 4.2. (Color figure online)

that define the skeleton disperses data points farther apart along the respective direction vectors. This primarily leads to a reduction of the id(X) of a cluster X, especially for smaller core numbers where singular movements have a bigger impact. For larger core numbers, the id(X) remains similar to the default settings as shown with *DENSIRED*$^{\delta}$ ($\delta = 2$; default: 1), though it is still reduced.

4 Experiments

4.1 Discussion of the Data Generator

Suitability for High Dimensionality
One of the effects of the *curse of dimensionality* is that pairwise distances become successively similar with increasing dimensionality [48]. Thus, a good data generator for density-based clusters should yield low pairwise intra-cluster distances for varying dimensionalities. In Fig. 4, we evaluate pairwise distances for points generated by various data generators with increasing dimensionality. The results are shown for three metrics: cosine, Euclidean, and density-connectivity distance (*dc-dist*) [49]. The effect is visualized for two widely used generating functions, *Blobs* and *Gaussian Quantiles* from scikit-learn, and for the generators MDCGen [33] and Seed Spreader [14]. Our method is shown with four different settings: i) *DENSIRED*: all parameters are set to their default values; ii) *DENSIRED*$^{\varkappa}$: star parameter set to $\varkappa = 1$; iii) *DENSIRED*$^{\beta}$: the branch parameter set to $\beta = 1$; iv) *DENSIRED*$^{\omega}$: the momentum factor ω set to 1; when varying one parameter, all other parameters are fixed to their default values. For the cosine metric, the expected behavior for Gaussian distributed clusters is that the average distance reaches 90 degrees when $n \to \infty$. With increasing dimensionality, our method preserves lower angles between the data points due to densely connected structures. The effect can also be observed for the Euclidean metric, for which, e.g., MDCGen, Blobs, and Gaussian Quantiles show degenerating effects on preserving low distances with increasing dimensionality, whereas the variants of the DENSIRED preserve lower distances in higher dimensional space. By construction, our method enriches data points in the skeleton such that data

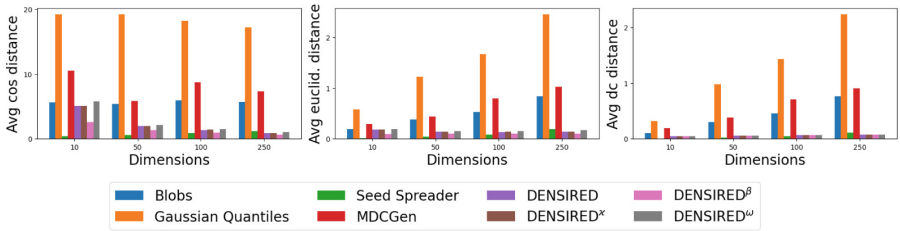

Fig. 4. Avg. pairwise distances of one cluster without noise for the metrics *cos*, *euclidean*, *dcdist* [49] for varying dimensionalities on datasets synthesized by *Blobs*, *Gaussian Quantiles* (sklearn), *MDCGen* [33], *Seed Spreader* [14] and variants of our generator: i) *DENSIRED*: default; ii) *DENSIRED*$^{\varkappa}$: $\varkappa = 1$; iii) *DENSIRED*$^{\beta}$: $\beta = 1$; iv) *DENSIRED*$^{\omega}$: $\omega = 1$; (all other parameters are fixed in each case). All datasets are scaled to a range between 0 and 1 based on their maximal values.

points are less likely extrema in higher dimensional space that would lead to a higher average Euclidean distance. By examining the structures with the dc-dist, the density-preserving characteristic, i.e., low intra-cluster distance values, is expected to be captured best, as both the metric and our data-generating function rely on the basic assumptions of density-connected core points. Our method can keep the lowest average dc-dist for a high dimensionality.

Impact of Parameters

The effect on a cluster's intrinsic dimensionality (id) on scaling the parameters *momentum* ω, *branch* β, *star* $\varkappa$, and *step size* δ is evaluated and summarized in Fig. 5. The evaluation shows the average results on 10 various seeds on a 100-dimensional dataset and we scale the number of generated cores in the range of $[0.1k, 0.5k, 1k]$. When scaling a specific parameter, the other ones are fixed to their default values. The y-axis shows the id with an explained variance of 0.99 of the input data. As shown on the left of Fig. 5, the id decreases with an increase of the momentum factor. Preserving the previous direction with a high momentum value for the random walk results in elongated clusters with low id. An increased number of cores increases the number of possibilities to elongate clusters in various directions, resulting in a slightly increased id for lower values for the momentum before it decreases as expected. As shown in the second part of Fig. 5, high values for β increase the id as they increase the restart probability during the random walk, making elongated clusters rather unlikely. As the *star* parameter $\varkappa$ indicates the behavior when restarting, it only has a marginal influence on the id for a low restart chance. Lastly, we evaluate the influence of the *step size* δ between a cluster's consecutive core points. As expected, when increasing the distance between the cores, the effect of the momentum's direction vector increases, which results in a lower id.

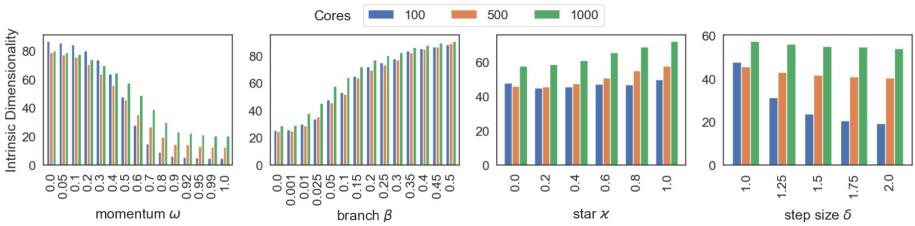

Fig. 5. Intrinsic dimensionality with a datasets's explained variance of 0.99 in $\mathbb{R}^{100}$ on scaling the parameters $\omega \in [0.0, 1.0]$, $\beta \in [0.0, 0.5]$, $\varkappa \in [0.0, 1.0]$, $\delta \in [1.0, 2.0]$ with *cores* $\in [0.1k, 0.5k, 1k]$ on 10 runs. When evaluating one parameter, others are set to default values: $\omega = 0.5$, $\beta = 0.05$, $\varkappa = 0.0$, $\delta = 1.0$. All datasets are scaled to a range between 0 and 1 based on their maximal values.

4.2 Benchmarking

Setup - Comptetitors + Dataset
For our benchmark study, we evaluated the following 14 clustering algorithms:

Centroid-Based Clustering. k-Means [42] minimizes the distances of points to their respective cluster centroids.

Spectral Clustering. Spectral Clustering [50] is based on the eigendecomposition of a similarity graph. SCAR [5] is an accelerated spectral clustering method. SpectACl [4] combines spectral and density-based clustering.

Hierarchical Clustering. Agglomerative clustering methods, like Ward's method [51], produce nested clusters in a bottom-up manner.

Deep Clustering. DEC [52] combines a clustering objective with neural networks. Unlike DEC, DipDECK [53] is capable of estimating the number of clusters within the latent space of an autoencoder.

Density-Based Clustering. Mean Shift [45] aims at finding modes of an underlying density function. The Density Peak Clustering Algorithm (DPCA) [54] finds clusters that are centered around a local density maximum. DCF [3] determines cluster cores for instances with highest densities based on kNN.

Density-Connectivity-Based Clustering. DBSCAN [2] finds density-connected clusters by connecting *core* points. OPTICS [7] is based on the concepts of DBSCAN and computes an augmented cluster-ordering. HDBSCAN [11] is an extension of DBSCAN that makes use of concepts from hierarchical clustering. MDD [9] can also identify clusters of different densities.

We used grid search to find the best hyperparameters for all algorithms. Where required, the number of clusters was set to number of synthesized ground truth clusters: k=10. The implementations for k-Means, Mean Shift, Spectral Clustering, Ward Clustering, DBSCAN and OPTICS stem from scikit-learn. The implementations for DEC, DipDeck and MDD [9] originate from ClustPy[2] [55].

[2] https://github.com/collinleiber/ClustPy, last accessed: Nov 15th, 2023.

Table 3. Clustering results for different dimensionalities

dimensionality	2-dim		5-dim		10-dim		50-dim		100-dim	
Metric	ARI	NMI	ARI	NMI	ARI	NMI	ARI	NMI	ARI	NMI
k-Means	0.50	0.62	0.32	0.52	0.65	0.74	0.61	0.75	0.61	0.79
Spectral Clust.	0.57	0.74	0.43	0.70	0.64	0.80	0.62	0.77	0.78	0.85
SCAR	0.60	0.69	0.47	0.65	0.58	0.75	0.71	0.78	0.84	0.86
SpectACl	0.60	0.66	0.54	0.80	0.69	0.87	0.78	0.87	0.85	0.92
Ward Clust.	0.43	0.59	0.35	0.57	0.61	0.75	0.66	0.80	0.72	0.85
DEC	0.36	0.51	0.41	0.61	0.72	0.85	0.59	0.77	0.52	0.75
DipDECK	0.52	0.62	0.34	0.52	0.66	0.76	0.75	0.82	0.77	0.86
Mean Shift	0.64	0.68	0.44	0.62	0.73	0.76	0.83	0.80	0.87	0.82
DPCA	0.43	0.58	0.19	0.41	0.61	0.72	0.77	0.80	0.82	0.85
DCF	0.91	0.90	1.00	0.99	1.00	0.99	0.95	0.88	0.95	0.88
DBSCAN	0.87	0.86	0.93	0.91	0.92	0.94	1.00	1.00	1.00	1.00
OPTICS	0.89	0.93	0.91	0.95	1.00	1.00	1.00	1.00	1.00	1.00
HDBSCAN	0.81	0.80	0.95	0.94	1.00	1.00	1.00	1.00	1.00	1.00
MDD	0.96	0.95	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

The implementation of HDBSCAN stems from the hdbscan python package[3]. DCF[4], SCAR[5], SpectACl[6], and DPCA[7] originate from the linked repositories.

The dataset contains branching data with high momentum and larger step size between consecutive cores[8], yielding high intrinsic dimensionalities and an elongated shape for each cluster. The two-dimensional dataset can be seen in Fig. 1 in the bottom-right. We generate datasets of varying dimensionality $d \in [2, 5, 10, 50, 100]$ scaled to a range between 0 and 1 based on their respective maximal values with 5000 data points each. We provide the datasets as novel benchmark datasets in our repository. We also include further evaluations on more datasets with different settings in our supplementary materials, which can be found in our GitHub repository.

Evaluation Metrics

We evaluate clustering results based on the ground truth labels with the commonly used Adjusted Rand Index (ARI) [56] and Normalized Mutual Information (NMI) [57]. ARI can take values $\in [-1, 1]$ and NMI lies in $[0, 1]$.

[3] https://github.com/scikit-learn-contrib/hdbscan, last accessed: Oct 5th, 2023.
[4] https://github.com/tobinjo96/DCFcluster, last accessed: Oct 5th, 2023.
[5] https://github.com/SpectralClusteringAcceleratedRobust/SCAR, last accessed: Nov 4th, 2023.
[6] https://bitbucket.org/Sibylse/spectacl, last accessed: Oct 11th, 2023.
[7] https://github.com/colinwke/dpca, last accessed: Oct 31st, 2023.
[8] Exact values: $\omega = 0.8$, $\delta = 1.5$, $\beta = 0.1$, $\varkappa = 1$, 200 cores.

Benchmarking Results
Table 3 summarizes the results and Fig. 3 showcases the general trend among the different clustering algorithms. Algorithms that support the usage of seeds were performed with 10 different seeds while keeping hyperparameter settings. k-Means cannot capture arbitrarily shaped clusters or noise. Ward Clustering is not robust against noise, and tends to fuse multiple clusters, which results in low performance scores. Spectral Clustering is generally sensitive to high noise levels in a dataset, as the decomposition of the affinity matrix does not yield distinct indicator vectors for various clusters. Hence, vanilla Spectral Clustering lacks in identifying the dense structures. SCAR has similar issues but manages to outperform Spectral Clustering. SpectACl faces similar difficulties for elongated clusters but is better at detecting dense structures in high-dimensional settings. DEC targets a latent representation optimizing the distributions of cluster assignments, which is not density-specific. DipDECK performs better as its micro-clusters are partially able to describe non-convex clusters in the latent space. Mean Shift often splits apart widespread clusters. Furthermore, the sensitivity of the bandwidth parameter in Mean Shift tends to misassign noisy data samples. DPCA tends to underestimate the size of the clusters and assigns large parts of clusters to noise (named cluster halo in [54]). This effect is more prominent for lower dimensionalities and for elongated clusters. Still, the method is robust against noise points. DCF performs reasonably well, but in areas of lower densities, where fewer core points overlap, DCF identifies subclusters. Noise in the dataset often produces individual density peaks, resulting in a lower clustering quality. DBSCAN introduced the concept of density-connectivity and succeeds at detecting the clusters. However, differing density levels exacerbate the choice of the right hyperparameters. Less dense clusters are often split into smaller clusters or are assigned to noise. Likewise to DBSCAN, OPTICS is highly dependent on the correct choice of a minimal number of samples to identify core points. Small values lead to significant parts of the clusters being considered as noise or subsumed into separate clusters, especially in lower dimensionalities. HDBSCAN subdivides less dense clusters and assigns noise to multiple smaller clusters. This effect is reduced in higher dimensionalities, inducing higher distances between noisy parts. MDD performs best out of all the evaluated algorithms. Notably, random noise is assigned to its own cluster for lower dimensionalities, whilst noisy points between clusters are correctly classified as noise.

In summary, due to the concept of density-connectivity, the class of density-based methods performs in general superior to other methods. While DBSCAN-related algorithms perform the best on the synthesized data, not all other density-based heuristics succeed in identifying the ground truth. Our benchmarking uncovers the necessity of proper data generators to enhance the modeling and evaluation of density-based clustering methods in a variety of settings.

5 Conclusion

A fair and reproducible comparison and evaluation of new algorithms is crucial for good research. While systematic testing is easy for data containing Gaussian

clusters, evaluation of density-based clustering algorithms was rather empirical in the past, as synthetic datasets were usually non-reproducible. In this paper, we present DENSIRED, a flexible and reliable data generator for data containing density-based clusters. It enables a systematic, reproducible, and controlled evaluation of all kinds of algorithms working on data that contain arbitrary density-connected shapes that are not generated by Gaussian processes. As the data generation process is separated into two steps, the model generation and the instantiation, it produces similar yet different datasets where users can define a multitude of properties for their specific use case. DENSIRED builds the basis for reproducible, fair, non-arbitrary comparison, evaluation, and benchmarking of data mining algorithms. The easy-to-use code as well as a set of benchmark datasets created with DENSIRED is publicly available in our repository.

References

1. Tommasi, T., Patricia, N., Caputo, B., Tuytelaars, T.: A deeper look at dataset bias. In: Csurka, G. (ed.) Domain Adaptation in Computer Vision Applications. ACVPR, pp. 37–55. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58347-1_2
2. Ester, M., Kriegel, H., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD, AAAI Press, pp. 226–231 (1996)
3. Tobin, J., Zhang, M.: DCF: an efficient and robust density-based clustering method. In: ICDM, pp. 629–638. IEEE (2021)
4. Hess, S., Duivesteijn, W., Honysz, P., Morik, K.: The SpectACl of nonconvex clustering: a spectral approach to density-based clustering. In: AAAI, AAAI Press, pp. 3788–3795 (2019)
5. Hohma, E., Frey, C.M.M., Beer, A., Seidl, T.: SCAR - spectral clustering accelerated and robustified. Proc. VLDB Endow. **15**(11), 3031–3044 (2022)
6. Sander, J., Ester, M., Kriegel, H., Xu, X.: Density-based clustering in spatial databases: the algorithm GDBSCAN and its applications. Data Min. Knowl. Discov. **2**(2), 169–194 (1998)
7. Ankerst, M., Breunig, M.M., Kriegel, H., Sander, J.: OPTICS: ordering points to identify the clustering structure, pp. 49–60 (1999)
8. Frey, C., Züfle, A., Emrich, T., Renz, M.: Efficient information flow maximization in probabilistic graphs. IEEE Trans. Knowl. Data Eng. **30**(5), 880–894 (2018)
9. Ashour, W., Sunoallah, S.: Multi density DBSCAN. In: Yin, H., Wang, W., Rayward-Smith, V. (eds.) IDEAL 2011. LNCS, vol. 6936, pp. 446–453. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23878-9_53
10. Frey, C.M., Jungwirth, A., Frey, M., Kolisch, R.: The vehicle routing problem with time windows and flexible delivery locations. Eur. J. Oper. Res. **308**(3), 1142–1159 (2023). ISSN 0377-2217
11. Campello, R.J.G.B., Moulavi, D., Sander, J.: Density-based clustering based on hierarchical density estimates. In: Pei, J., Tseng, V.S., Cao, L., Motoda, H., Xu, G. (eds.) PAKDD 2013. LNCS (LNAI), vol. 7819, pp. 160–172. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37456-2_14
12. Yale, A., Dash, S., Dutta, R., Guyon, I., Pavao, A., Bennett, K.P.: Generation and evaluation of privacy preserving synthetic health data. Neurocomputing **416**, 244–255 (2020). ISSN 0925-2312

13. Libes, D., Lechevalier, D., Jain, S.: Issues in synthetic data generation for advanced manufacturing. In: 2017 IEEE International Conference on Big Data (Big Data), Boston, MA, USA, pp. 1746-1754 (2017)
14. Gan, J., Tao, Y.: DBSCAN revisited: Mis-claim, un-fixability, and approximation. In: SIGMOD Conference, pp. 519–530. ACM (2015)
15. Mai, S.T., Assent, I., Storgaard, M.: AnyDBC: an efficient anytime density-based clustering algorithm for very large complex datasets. In: KDD, pp. 1025–1034. ACM (2016)
16. Hou, J., Gao, H., Li, X.: DSets-DBSCAN: a parameter-free clustering algorithm. IEEE Trans. Image Process. **25**(7), 3182–3193 (2016)
17. Bryant, A., Cios, K.J.: RNN-DBSCAN: a density-based clustering algorithm using reverse nearest neighbor density estimates. IEEE Trans. Knowl. Data Eng. **30**(6), 1109–1121 (2018)
18. Kim, J., Choi, J., Yoo, K., Nasridinov, A.: AA-DBSCAN: an approximate adaptive DBSCAN for finding clusters with varying densities. J. Supercomput. **75**(1), 142–169 (2019)
19. Ren, Y., Wang, N., Li, M., Xu, Z.: Deep density-based image clustering. Knowl. Based Syst. **197**, 105841 (2020)
20. Chen, Y., Zhou, L., Bouguila, N., Wang, C., Chen, Y., Du, J.: BLOCK-DBSCAN: fast clustering for large scale data. Pattern Recognit. **109**, 107624 (2021)
21. dos Santos, J.A., Iqbal, S.T., Naldi, M.C., Campello, R.J.G.B., Sander, J.: Hierarchical density-based clustering using MapReduce. IEEE Trans. Big Data **7**(1), 102–114 (2021)
22. Wang, Z., et al.: AMD-DBSCAN: an adaptive multi-density DBSCAN for datasets of extremely variable density. In: DSAA, pp. 1–10. IEEE (2022)
23. Huang, X., Ma, T., Liu, C., Liu, S.: GriT-DBSCAN: a spatial clustering algorithm for very large databases. Pattern Recognit. **142**, 109658 (2023)
24. Ma, B., Yang, C., Li, A., Chi, Y., Chen, L.: A faster dbscan algorithm based on self-adaptive determination of parameters. Procedia Comput. Sci. **221**, 113–120 (2023). (ITQM 2023)
25. Qian, J., Zhou, Y., Han, X., Wang, Y.: MDBSCAN: a multi-density dbscan based on relative density. Neurocomputing **576**, 127329 (2024)
26. Milligan, G.W.: An algorithm for generating artificial test clusters. Psychometrika **50**, 123–127 (1985)
27. Qiu, W., Joe, H.: Generation of random clusters with specified degree of separation. J. Classif. **23**(2), 315–334 (2006)
28. Melnykov, V., Chen, W.-C., Maitra, R.: MixSim: an r package for simulating data to study performance of clustering algorithms. J. Stat. Softw. **51**, 1–25 (2012)
29. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
30. Fachada, N., de Andrade, D.: Generating multidimensional clusters with support lines. Knowl. Based Syst. **277**, 110836 (2023)
31. Steinley, D.L., Henson, R.: OCLUS: an analytic method for generating clusters with known overlap. J. Classif. **22**(2), 221–250 (2005)
32. Shand, C., Allmendinger, R., Handl, J., Webb, A.M., Keane, J.: HAWKS: evolving challenging benchmark sets for cluster analysis. IEEE Trans. Evol. Comput. **26**(6), 1206–1220 (2022)
33. Iglesias, F., Zseby, T., Ferreira, D.C., Zimek, A.: MDCGen: multidimensional dataset generator for clustering. J. Classif. **36**(3), 599–618 (2019)

34. Vennam, J.R., Vadapalli, S.: SynDECA: a tool to generate synthetic datasets for evaluation of clustering algorithms. In: COMAD, Computer Society of India, pp. 27–36 (2005)
35. Gan, J., Tao, Y.: On the hardness and approximation of euclidean DBSCAN. ACM Trans. Database Syst. **42**(3), 14:1–14:45 (2017)
36. Rachkovskij, D.A., Kussul, E.M.: DataGen: a generator of datasets for evaluation of classification algorithms. Pattern Recognit. Lett. **19**(7), 537–544 (1998)
37. Fränti, P., Sieranoja, S.: K-means properties on six clustering benchmark datasets, pp. 4743–4759 (2018). http://cs.uef.fi/sipu/datasets/
38. Beer, A., Schüler, N.S., Seidl, T.: A generator for subspace clusters. In: LWDA, ser. CEUR Workshop Proceedings, vol. 2454, pp. 69–73 (2019). CEUR-WS.org
39. Schubert, E., Sander, J., Ester, M., Kriegel, H., Xu, X.: DBSCAN revisited, revisited: why and how you should (still) use DBSCAN. ACM Trans. Database Syst. **42**(3), 19:1–19:21 (2017)
40. Goncalves, A., Ray, P., Soper, B., Stevens, J., Coyle, L., Sales, A.P.: Generation and evaluation of synthetic patient data. BMC Med. Res. Methodol. **20**(1), 1–40 (2020)
41. Pei, Y., Zaiane, O.R.: A synthetic data generator for clustering and outlier analysis (2006)
42. Lloyd, S.P.: Least squares quantization in PCM. IEEE Trans. Inf. Theory **28**(2), 129–136 (1982)
43. Georgoulas, G.K., Konstantaras, A., Katsifarakis, E., Stylios, C.D., Maravelakis, E., Vachtsevanos, G.J.: "Seismic-mass" density-based algorithm for spatio-temporal clustering. Expert Syst. Appl. **40**(10), 4183–4189 (2013)
44. Jain, A.K.: Data clustering: 50 years beyond k-means. Pattern Recognit. Lett. **31**(8), 651–666 (2010)
45. Comaniciu, D., Meer, P.: Mean Shift: a robust approach toward feature space analysis. IEEE Trans. Pattern Anal. Mach. Intell. **24**(5), 603–619 (2002)
46. Ullmann, T., Beer, A., Hünemörder, M., Seidl, T., Boulesteix, A.: Over-optimistic evaluation and reporting of novel cluster algorithms: an illustrative study. Adv. Data Anal. Classif. **17**(1), 211–238 (2023)
47. Levina, E., Bickel, P.: Maximum likelihood estimation of intrinsic dimension. In: Saul, L., Weiss, Y., Bottou, L. (eds.) NIPS, vol. 17. MIT Press (2004)
48. Beyer, K., Goldstein, J., Ramakrishnan, R., Shaft, U.: When Is "Nearest Neighbor" meaningful? In: Beeri, C., Buneman, P. (eds.) ICDT 1999. LNCS, vol. 1540, pp. 217–235. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-49257-7_15
49. Beer, A., Draganov, A., Hohma, E., Jahn, P., Frey, C.M., Assent, I.: Connecting the dots - density-connectivity distance unifies dbscan, k-center and spectral clustering. In: KDD, pp. 80–92. ACM (2023)
50. von Luxburg, U.: A tutorial on spectral clustering. Stat. Comput. **17**(4), 395–416 (2007)
51. Ward, J.H., Jr.: Hierarchical grouping to optimize an objective function. J. Am. Stat. Assoc. **58**(301), 236–244 (1963)
52. Xie, J., Girshick, R.B., Farhadi, A.: Unsupervised deep embedding for clustering analysis. In: ICML, ser. JMLR Workshop and Conference Proceedings, vol. 48, pp. 478–487 (2016). JMLR.org
53. Leiber, C., Bauer, L.G.M., Schelling, B., Böhm, C., Plant, C.: Dip-based deep embedded clustering with k-estimation. In: KDD, pp. 903–913. ACM (2021)
54. Rodriguez, A., Laio, A.: Clustering by fast search and find of density peaks. Science **344**(6191), 1492–1496 (2014)

55. Leiber, C., Miklautz, L., Plant, C., Böhm, C.: Benchmarking deep clustering algorithms with clustpy. In: ICDM (Workshops), pp. 625–632. IEEE (2023)
56. Hubert, L., Arabie, P.: Comparing partitions. J. Classif. **2**, 193–218 (1985)
57. Strehl, A., Ghosh, J.: Cluster ensembles – a knowledge reuse framework for combining multiple partitions. J. Mach. Learn. Res. **3**, 583–617 (2002)