Supplementary material for: 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. christianmaxmike@gmail.com
[4] University of Vienna, Vienna, Austria. anna.beer@univie.ac.at

The supplement expands on the related work included in "Data with Density-Based Clusters: A Generator for Systematic Evaluation of Clustering Algorithms" in Appendix A. This includes a more in-depth look at the other data generators in Appendix A.1 and a detailed description of the clustering algorithms used in Appendix A.2.
The supplement also gives additional explanations on the *equivalence* of datasets in Appendix B.
Furthermore, we include results and background for an extended benchmark study with 6 additional algorithms and an additional basic data setting in Appendix C. All datasets, including the 2d visualization dataset, as well as the experiment logs for this evaluation, can be found in the GitHub repository of the paper: https://github.com/PhilJahn/DENSIRED.

A Expanded Related Work

A.1 Data Generators

There are data generators for many different settings [1], e.g., for multivariate ordinal data [2], for subspace clustering [3], and for text, trees, and logical interpretations [4]. Thus, data-generating functions have been established as a useful tool for a controlled evaluation of procedures in a plethora of domains ranging from healthcare [5, 6], routing problems [7], flow networks [8], to manufacturing [9].
In [10], the authors use a "self-implemented data generator" as the basis for their experiments for evaluating a novel density-based subspace clustering. However, neither implementation nor exact specifics that are sufficient to reproduce the datasets are given.
Pei's data generator described in [11] offers the ability to create clusters of different shapes based on difficulty levels. This includes clusters that could be suitable for evaluating density-based clustering algorithms. However, as the approach is

limited to two-dimensional data, it can not be used to evaluate a method's quality in higher dimensional space.

Milligan's method introduced in [12,13] sets up cluster boundaries and fills them based on a random distribution around the midpoints of the boundary lengths. This approach was augmented in [14], adding the ability to specify the degree of separation and allowing arbitrary positive definite matrices as covariance matrices. Nonetheless, the data is limited to what can be generated by a distribution around a singular midpoint.

MixSim [15] is a popular data generator for overlapping clusters. It simulates Gaussian mixture models with varying covariance matrices. Using an inverse Box-Cox transformation, MixSim can also provide non-normally distributed data. However, MixSim only uses a single Gaussian distribution per component and clusters' shapes are limited by the features of inverse Box-Cox transformations. This, in turn, limits MixSim to non-arbitrary shapes. Still, MixSim has been used to evaluate density-based clustering in [16].

Scikit-learn [17] offers several ways to generate data, though most aim towards 2-dimensional data. Primarily of interest are the blobs and swiss-roll generators, which have been used to evaluate density connectivity-based clustering methods [18]. The blobs generator produces isotropic Gaussian distributions that are either randomly placed or have their center specified by user input. However, the generator is limited to purely Gaussians, and can thus not create arbitrarily shaped clusters. The swiss-roll generator is limited to 3 dimensions and does not generate separate ground-truth clusters but rather a single distribution that is shaped like a roll.

Clugen [19] uses line segments as a basis for generating clusters with directionality. It builds on top of a 2-dimensional data generator [20] but expands the method to a multi-dimensional setting. Clugen is based on arbitrary statistical distributions. As clusters have only a single supporting line, their overall shape can not be arbitrarily shaped.

OCLUS [21] offers a data generation approach for overlapping clusters, expressed by shared density between clusters. The clusters are based on user-defined random distributions, but as these distributions only apply independently in distinct dimensions, clusters are constrained to be of non-arbitrary shape.

HAWKS [22] is a data generation method based on an evolutionary algorithm. HAWKS creates Gaussian clusters where the distributions are adjusted based on mutation and cross-over from parent datasets. Constraints can be applied that are used to select these parents, in order to target specific properties for the final dataset. Still, Gaussian clusters can not be arbitrarily shaped.

MDCGEN [23] is a data generator for subspace clusters. It uses hyper-grids for overlap control and cluster positioning and can produce radial-based or multivariate clusters. The coordinates of a cluster's points are drawn from a distribution for each dimension. This includes ring-shaped distributions that can be considered density-connected. However, this leads to only a very limited range of density-connected clusters, which is not sufficient for a broad systematic evaluation of density-based clustering algorithms: the crucial benefits of these clustering methods can primarily be seen and evaluated on arbitrarily shaped clusters, which can not be generated with MDCGEN.

SynDECA [24] offers to create clusters of different shapes. Before creating the actual clusters, bounding boxes are determined to prevent any overlap. SynDECA randomly chooses a shape type for each bounding box. Here, the irregular shapes are especially interesting. SynDECA offers two ways of generating irregular shapes. Irregular 1 performs a type of random walk that expands a random data point with an ε-range by further points with the same radius. This could, however, violate the bounding box and overlap with other clusters. Furthermore, this approach offers very little control. Irregular 2 works by splitting the bounding box into smaller boxes. Starting from a random box, an adjacent box is selected and randomly filled with points. This continues until a threshold of data points is reached. While this produces complex-shaped clusters, the usage of hyperrectangular bounding boxes makes the clusters easily separable despite the actual shape.

DataGen [25] is a data generator that first generates class boundaries based on an initial random assignment of data points which serve as training data for a classifier. The decision boundaries of this classifier are used to define class regions which are then populated with data points following a random distribution. While powerful, DataGen is limited by the decision boundaries of the classifier; there are no guarantees that regions are continuously dense or even connected. Furthermore, users can not control most properties of the data.

Seed Spreader [26, 27] was already used [26, 28] to create higher-dimensional density-connected clusters. It works by generating data points with a fixed radius (based on either a global radius value or on the current cluster number [27]) surrounding the current position of the random walk. It either samples until a predefined amount of points within the hypersphere is reached or - based on a preset chance - jumps to a random position in the data space, at which the hypersphere sampling is terminated early. If the full amount of points has been generated from the hypersphere, the random walk moves on. Seed Spreader has multiple issues that DENSIRED addresses (i.e., a lack of user control, potentially overlapping clusters, and no knowledge about the number of clusters prior to the generation), which are covered in the main paper.

Availability Availability of a generator is a very important aspect for reproducible evaluation and benchmarking.

Fachada and de Andrade [19] already noted this importance in their paper on Clugen. They made Clugen available in multiple programming languages[5] and also documented the availability of related data generators.

Pei's generator [11] is, to the best of our knowledge, not publicly available [19]. An implementation of Milligan's method [12] is available as a C++ implementation [19, 29][6], but the implementation is not usable out of the box [19].

The improved version of Milligan's method [14] is available as an R implemen-

[5] `https://github.com/clugen`, last accessed: Mar 22nd, 2024

[6] `https://clusutils.sourceforge.net/`, last accessed: Mar 22nd, 2024

tation [19][7].
MixSim [15] is also available as an R implementation [15, 19][8].
The Scikit-learn library is public, and as such both the blobs as well as the swiss-roll generators are easily accessible in Python[9].
OCLUS [21] is, to the best of our knowledge, not publicly available, though its code can be requested from the authors [19].
A Python implementation of HAWKS [22] is available [19, 22][10].
MDCGen is available in both Python and MATLAB [19, 23][11].
SynDECA appears to have been available at some point [19][12], but the link is no longer functional, making the method inaccessible.
The authors of DataGen [25] expressed their intention of making their code available in the UCI repository [30], but we have not been able to find the generator.
To the best of our knowledge, Seed Spreader was not made publicly available alongside the publications describing it [26, 27]. However, our Python reimplementation based on the provided description can be found in our repository.
Our method, DENSIRED, is available there as a Python implementation as well.

A.2 Clustering Methods

We introduce the clustering methods used in the expanded benchmarking in Appendix C in the following.

Centroid-based Clustering k-Means [31] is one of the most fundamental clustering algorithms and requires only the number of clusters as user input. Its goal is to minimize the distances of all points to their respective cluster centroids, which is most often done in an iterative, non-deterministic way.
LDA-Km [32] is an extension of k-Means that incorporates linear discriminant analysis (LDA) [33]. LDA is used to select a suitable subspace for clustering, while the clustering itself is performed by k-Means. As both methods optimize the same objective function, subspace and cluster adjustments are performed simultaneously.
X-Means [34, 35] is an adaption of k-Means that integrates information-theoretic criteria to maximize the model's quality. The major advantage besides its efficiency is that X-Means finds the number of clusters automatically.

[7] `https://cran.r-project.org/web/packages/clusterGeneration/index.html`, last accessed: Mar 22nd, 2024

[8] `https://cran.r-project.org/web/packages/MixSim/index.html`, last accessed: Mar 22nd, 2024

[9] `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_blobs.html` and `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_swiss_roll.html`, last accessed: Mar 22nd, 2024

[10] `https://github.com/sea-shunned/hawks`, last accessed: Mar 22nd, 2024

[11] `https://github.com/CN-TU/mdcgenpy` and `https://github.com/CN-TU/mdcgen-matlab`, last accessed: Mar 22nd, 2024

[12] `https://sites.google.com/site/syndeca/`, last attempted access: Mar 22nd, 2024

Spectral Clustering

Spectral Clustering [36–38] is another fundamental clustering concept. It is based on the eigendecomposition of a similarity graph representing the data. Points are then clustered in their spectral embedding, where often k-Means is used.

SCAR [39] is a novel spectral clustering method working on weighted kNN graphs. It removes noisy edges from the affinity graph building on top of [40] while accelerating the eigendecomposition with the Nyström method [41].

SpectACl [42] combines spectral and density-based clustering by maximizing the average cluster density. It is based on the ratio cut of the normalized ϵ-neighborhood graph given by its spectrum.

Hierarchical Clustering Agglomerative clustering methods like [43–46] are bottom-up hierarchical clustering methods that are based solely on some similarity measure between points. They differ mainly in the way they measure similarity between groups of points, where single-linkage, average-linkage, complete-linkage, and Ward's method [43] are the most common methods. Ward was the method chosen for the evaluation in Appendix C (as well as for the main paper).

Birch [47] is a hierarchical clustering approach that is based on a height-balanced tree with so-called *Clustering Features* as leaves. Areas of higher density are considered (nested) subclusters, while points in sparse areas are removed as outliers. Clustering features contain information on the number of data points as well as the linear and squared sums of their values. Each non-leaf node corresponds to a higher-level cluster containing its children.

Deep Clustering

In recent years, deep clustering approaches that combine a clustering objective with neural networks have become increasingly popular. DEC [48] works by simultaneously learning a feature representation of the data and their cluster assignments using the Kullback-Leibler divergence. It is initialized as a stacked autoencoder, but only the encoder part is retained to produce the feature embedding for clustering.

IDEC [49] is an extension of DEC that keeps the decoder part of the autoencoder to retain the local structure and to ensure that the representation is not distorted. The clustering loss that is used to obtain an embedding conductive of clustering is directly applied to the embedding, whereas the decoder is untouched and has to be able to reconstruct the data again.

Unlike DEC and IDEC, DipDECK [50] is capable of estimating the number of clusters within the latent space of an autoencoder. It does so by using Hartigans' dip test [51] after heavily overestimating the number of clusters.

The DipEncoder [52] uses a gradient formulation of Hartigans' dip values [51] to separate multimodal data in the embedded space. The cluster assignment is achieved on the so-called modal interval, which defines the main mode of the data.

Density-based Clustering

Mean Shift [53] aims at finding modes of an underlying density function by recursively applying the eponymous mean shift procedure [54, 55]. A cluster contains the data points visited by all the mean shift procedures after they converged to the same location. Data points visited by multiple procedures are assigned to the predominant one.

The Density Peak Clustering Algorithm (DPCA) [56] finds clusters that are centered around a local density maximum by regarding the distances of points to the closest point of higher density. Cluster assignments are based on the cluster of the nearest neighbor with higher density.

DCF [57] uses the concept of modal sets introduced in [58]. It determines cluster cores for instances with the highest densities based on kNN with similar densities. This leads to separated cluster cores that imply the clusters.

Data Density Based Clustering (DDC) [59] uses the highest density points as cluster centers and assigns data points within a cluster radius. Clusters are refined and already assigned samples are removed before repeating the process until the number of considered points falls below a threshold.

Density-Connectivity-based Clustering

DBSCAN [26, 60, 61] finds density-connected clusters of arbitrary shape by connecting *core* points (i.e., points that have at least *minPts* neighbors in their ε-range) in a manner similar to Prim's algorithm. It finds the number of clusters automatically and considers noise points.

OPTICS [62, 63] is based on the concepts of DBSCAN and computes an augmented cluster ordering usually visualized as OPTICS plot. This plot allows users to easily find a suitable ε for DBSCAN as well as a DBSCAN-like clustering [62] and gain insights into the cluster structure. In our experiments in Appendix C, OPTICS clusters were automatically obtained from OPTICS plot based on the ξ-steepness [62].

HDBSCAN [64,65] is an extension of DBSCAN that makes use of concepts from hierarchical clustering. The core idea of HDBSCAN is to identify clusters that persist over multiple ε-values, i.e., multiple density levels. The hierarchy of the underlying tree structure of HDBSCAN is determined by the clusters' stability over varying ε-values as well as a minimal cluster size. HDBSCAN aims at optimizing the stability of all clusters while maintaining their disjointness.

Multi-density DBSCAN (MDD) [66] extends DBSCAN to deal with clusters of different densities. It iteratively processes data points with decreasing density and forms clusters by expanding to the k-neighbors with similar densities.

B Expanded Theory, and Concept

B.1 "Density-connectivity"

Our generator creates datasets containing density-connected clusters. We follow the well-known concept from clustering algorithms like DBSCAN [60] or

HDBSCAN [64]. Note that this differs from mode-seeking density-based concepts as, e.g., used in Mean Shift [53]. For completeness, we provide definitions for density-connectivity based on [60, 67].

Definition 1. *ε-neighborhood and core-property*
The ε-neighborhood of a point p in a dataset X is defined as $N_\varepsilon(p) = \{q \in X : dist(p, q) \leq \varepsilon\}$ for $\varepsilon > 0$ and a distance measure $dist(\cdot, \cdot)$. 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.

Definition 2. *(Directly) Density-Reachable*
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.
A point q is density-reachable from a point p iff there is a chain of directly density-connected core points between them.

Definition 3. *Density-Connected*
A point p is density-connected to a point q for $\varepsilon > 0$, $\mu \in \mathbb{N}$ iff there is a point o such that p and q are both density-reachable from o.

Definition 4. *Density-based Cluster*
A (density-based) cluster $C^{\varepsilon,\mu} \subseteq X$ is a "maximal set of density-connected objects" [67].

B.2 Equivalence of datasets

For analyzing centroid-based clustering algorithms, clusters are usually generated by drawing from a Gaussian distribution with predefined centers and variances. The resulting datasets are similar, but not equal. With that intuition, we call such clusters equivalent and define it accordingly for density-based clusterings:

Definition 5. *Equivalence of (density-based) clusterings.*
Two density-based clusterings $\mathcal{A}, \mathcal{B}$ are equivalent if there is a bijective mapping between them. I.e., there is a bijective mapping $f_c : \mathcal{A} \to \mathcal{B}$ of the clusters such that for every cluster in $\mathcal{A}$, there is also a bijective mapping of all its points to the points in the associated cluster in $\mathcal{B}$: $\forall c^{\mathcal{A}} \in \mathcal{A}\ \exists c^{\mathcal{B}} : f_p : c^{\mathcal{A}} \to c^{\mathcal{B}}$.

Definition 6. *Equivalence of density-based datasets.*
Two datasets $X \in \mathbb{R}^{n \times d}, Y \in \mathbb{R}^{n \times d}$ are equivalent regarding their density-connectivity if, for every ε and μ, there exist δ, ν, such that the ε, μ- clustering of X is equivalent to the δ, ν-clustering of Y.

For an unbiased evaluation of a novel method, it is important to eliminate random effects. Especially for datasets with density-based clusters, these effects can be decisive for the perceived quality of, e.g., a clustering model, see [68]. Using several equivalent datasets allows the elimination of these random effects for a proper analysis and evaluation of novel clustering algorithms and other methods working on similar data.

B.3 Challenge: Generating equivalent clusters

Generating equivalent density-based clusters is a challenging problem. In contrast to centroid-based clustering concepts, density-based clustering has a discrete, non-continuous aspect expressed by the parameter $\mu \in \mathbb{N}$. Thus, changing a single point in a dataset can lead to a drastically different clustering. Especially, clusters that fulfill the density-criterion only tightly can disintegrate into several clusters if a decisive point is missing or changing positions as shown in Figure 1.

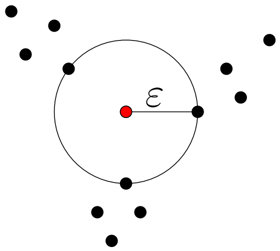

Fig. 1: Consider this dataset and its clustering for $\mu = 4$ and ε as given by the radius of the circle. All data points form one cluster. If the central red data point is moved to any other position, the cluster splits up. Without the red data point, there are three density-based clusters in the dataset.

C Expanded Benchmarking

C.1 Experiment Setup

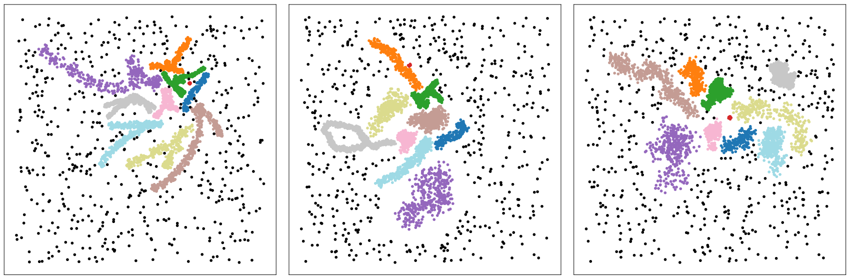

Fig. 2: Two-dimensional benchmark datasets (the cluster shape is more 'stretched' and the right is 'compact'; the middle 'mix' data was only used for the two-dimensional case).

For the expanded benchmark study, we evaluated 20 clustering algorithms on data produced by DENSIRED. For an in-depth analysis of higher-dimensional space, we evaluate the models on two different basic settings with a shared set of parameters besides the dimensionality:
i) The first category ("stretched") contains branching data with high momentum and larger step size between consecutive cores[13], yielding more spread out clusters with tendentially higher intrinsic dimensionalities. This setting was also the one employed in the main paper.
ii) The second category ("compact") contains non-branching data without momentum resulting in compact cluster structures[14].
The two-dimensional versions for both settings are shown in Figure 2 on the left and right sides. Figure 3 features the corresponding cluster skeletons for the two-dimensional case. For both settings, 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 for each dataset.

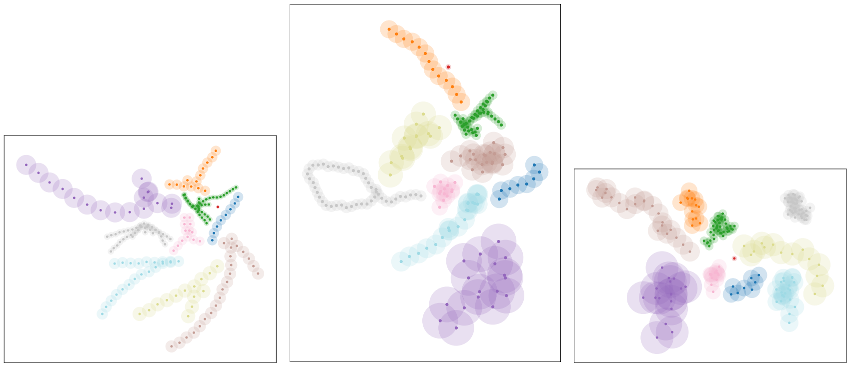

Fig. 3: Skeletons of the two-dimensional benchmark datasets (the cluster shape is more 'stretched' and the right is 'compact'; the middle 'mix' data was only used for the two-dimensional case).

Note that the visualized data represents just a small subset of the generator's capabilities. Aside from the data generated for the "stretched" and "compact" settings, there is also a third 2D dataset "mix", which features a more varied setting of the parameters but was primarily intended for visualization purposes. It is displayed in the middle for both Figure 2 and Figure 3. The datasets used in our experiments are provided as novel benchmark datasets in our repository. We conducted extensive parameter grid searches to find the best possible quality for each model for a fair comparison. In cases where an algorithm requires an ex-

[13] Exact values: $\omega = 0.8$, $\delta = 1.5$, $\beta = 0.1$, $\varkappa = 1$, 200 cores

[14] Exact values: $\omega = 0$, $\delta = 1$, $\beta = 0$, $\varkappa = 0$, 200 cores

plicit number of clusters, the number of clusters was set to 10 in correspondence to the number of synthesized (density-based) ground-truth clusters. For simplicity, the results of the best-performing setting for each algorithm were chosen and are used here (best mean performance for both "compact" and "stretched", highest clustering performance for "mix"). Best performing was defined as the highest sum of ARI [69] and NMI [70]. A full log of the results for the evaluated parameters in the initial parameter sweep is listed in our repository.

The implementations for k-Means, Mean Shift, Spectral Clustering, Agglomerative Clustering, BIRCH, DBSCAN and OPTICS stem from scikit-learn[15] [17]. The implementations for LDA-Km, X-Means, DEC, IDEC, DipDeck, DipEncoder and MDD originate from ClustPy[16] [71]. The implementation of HDBSCAN stems from the hdbscan python package[17]. DCF[18], SCAR[19], SpectACl[20], DPCA[21], and DDC[22] originate from the linked repositories.

C.2 "Stretched"-Setting

The results for the "stretched"-setting for the data generator are listed in Table 1. Figure 4 shows the general tendencies in performance of the algorithms across all examined dimensionalities. The best-performing parameters are described in Table 2.

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

[16] `https://github.com/collinleiber/ClustPy/tree/main`, last accessed: November 15th, 2023; LDA-Km, X-Means and MDD used the initial v0.0.1-alpha version; DEC, IDEC, DipDeck and DipEncoder used v0.0.2-beta

[17] `https://github.com/scikit-learn-contrib/hdbscan/tree/master`, last accessed: October 5th, 2023

[18] `https://github.com/tobinjo96/DCFcluster`, last accessed: October 5th, 2023

[19] `https://github.com/SpectralClusteringAcceleratedRobust/SCAR`, last accessed: November 4th, 2023

[20] `https://bitbucket.org/Sibylse/spectacl/src/master/`, last accessed: October 11th, 2023

[21] `https://github.com/colinwke/dpca`, last accessed: October 31st, 2023

[22] `https://github.com/RHyde67/DDC-Python`, last accessed: November 16th, 2023

Table 1: Clustering results for different dimensionalities for the "stretched"-Setting

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 ±0.04	0.62 ±0.02	0.32 ±0.02	0.52 ±0.01	0.65 ±0.04	0.74 ±0.03	0.61 ±0.04	0.75 ±0.02	0.61 ±0.05	0.79 ±0.02
LDA-Km	0.48 ±0.03	0.61 ±0.02	0.31 ±0.01	0.52 ±0.01	0.60 ±0.06	0.71 ±0.04	0.57 ±0.03	0.73 ±0.02	0.67 ±0.00	0.80 ±0.00
X-Means	0.53 ±0.02	0.63 ±0.01	0.35 ±0.02	0.64 ±0.02	0.63 ±0.03	0.74 ±0.02	0.72 ±0.01	0.80 ±0.01	0.83 ±0.05	0.86 ±0.02
Spectral Clust.	0.57 ±0.00	0.74 ±0.00	0.43 ±0.00	0.70 ±0.00	0.64 ±0.00	0.80 ±0.00	0.62 ±0.02	0.77 ±0.01	0.78 ±0.00	0.85 ±0.00
SCAR	0.60 ±0.02	0.69 ±0.01	0.47 ±0.01	0.65 ±0.02	0.58 ±0.00	0.75 ±0.00	0.71 ±0.02	0.78 ±0.02	0.84 ±0.00	0.86 ±0.00
SpectACl	0.60 ±0.04	0.66 ±0.02	0.54 ±0.04	0.80 ±0.02	0.69 ±0.02	0.87 ±0.01	0.78 ±0.02	0.87 ±0.01	0.85 ±0.04	0.92 ±0.02
Ward Clust.	0.43	0.59	0.35	0.57	0.61	0.75	0.66	0.80	0.72	0.85
Birch	0.49	0.61	0.27	0.49	0.61	0.74	0.66	0.80	0.72	0.85
DEC	0.36 ±0.06	0.51 ±0.05	0.41 ±0.02	0.61 ±0.02	0.72 ±0.05	0.85 ±0.03	0.59 ±0.08	0.77 ±0.04	0.52 ±0.06	0.75 ±0.05
IDEC	0.43 ±0.07	0.55 ±0.05	0.40 ±0.04	0.60 ±0.03	0.72 ±0.06	0.85 ±0.03	0.64 ±0.08	0.80 ±0.05	0.73 ±0.06	0.83 ±0.03
DipDECK	0.52 ±0.06	0.62 ±0.04	0.34 ±0.03	0.52 ±0.03	0.66 ±0.06	0.76 ±0.03	0.75 ±0.05	0.82 ±0.03	0.77 ±0.09	0.86 ±0.04
DipEncoder	0.42 ±0.04	0.60 ±0.02	0.42 ±0.05	0.57 ±0.02	0.73 ±0.03	0.84 ±0.02	0.66 ±0.05	0.79 ±0.02	0.69 ±0.10	0.83 ±0.06
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
DDC	0.42	0.64	0.34	0.63	0.63	0.72	0.65	0.69	0.78	0.69
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
MD-DBSCAN	0.96	0.95	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

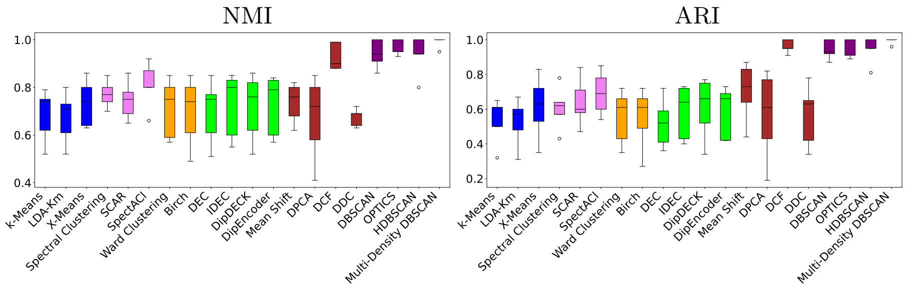

Fig. 4: NMI (left) and ARI (right) across all examined dimensionalities for the setting from the "stretched" setting. The coloration indicates the algorithm category as described in Appendix A.2.

C.3 "Compact"-Setting

The results for the "stretched"-setting for the data generator are listed in Table 3. Figure 5 shows the general tendencies in performance of the algorithms across all examined dimensionalities. The best-performing parameters are described in Table 4.

C.4 "Mix"-Setting

The "mix"-setting did not include any higher-dimensional datasets. It was primarily intended for visualization purposes. The cluster assignments for the best-performing clustering runs are displayed in Figure 6, where best-performing means the highest sum of NMI and ARI. The best-performing parameters are described in Table 5.

Table 2: Best performing parameters (out of the evaluated ones) for different dimensionalities for the "stretched"-Setting

dimensionality	2-dim	5-dim	10-dim	50-dim	100-dim
k-Means	n_clusters: 10, n_init: 2	n_clusters: 10, n_init: 2	n_clusters: 10, n_init: 3	n_clusters: 10, n_init: 7	n_clusters: 10, n_init: 6
LDA-Km	n_clusters: 10, n_init: 1	n_clusters: 10, n_init: 1	n_clusters: 10, n_init: 1	n_clusters: 10, n_init: 2	n_clusters: 10, n_init: 6
X-Means	n_clusters_init: 8, max_n_clusters: 10, n_split_trials: 3	n_clusters_init: 6, max_n_clusters: 28, n_split_trials: 8	n_clusters_init: 7, max_n_clusters: 10, n_split_trials: 1	n_clusters_init: 3, max_n_clusters: 10, n_split_trials: 2	n_clusters_init: 2, max_n_clusters: 17, n_split_trials: 10
Spectral Clust.	n_clusters: 10, affinity: 'nearest_neighbors', nearest_neighbors: 25	n_clusters: 10, affinity: 'nearest_neighbors', nearest_neighbors: 75	n_clusters: 10, affinity: 'nearest_neighbors', nearest_neighbors: 120	n_clusters: 10, affinity: 'nearest_neighbors', nearest_neighbors: 140	n_clusters: 10, affinity: 'nearest_neighbors', nearest_neighbors: 95
SCAR	k: 10, nn: 15, alpha: 0.9, theta: 1000 weighted: True, laplacian: 0, normalize: True	k: 10, nn: 15, alpha: 0.9, theta: 2000 weighted: True, laplacian: 0, normalize: True	k: 10, nn: 100, alpha: 0.9, theta: 2000 weighted: True, laplacian: 0, normalize: True	k: 10, nn: 100, alpha: 0.7, theta: 3000 weighted: True, laplacian: 0, normalize: True	k: 10, nn: 50, alpha: 0.8, theta: 2000 weighted: True, laplacian: 0, normalize: True
SpectACl	n_clusters: 10, affinity: 'radius_neighbors', epsilon: 0.12	n_clusters: 10, affinity: 'radius_neighbors', epsilon: 0.04	n_clusters: 10, affinity: 'radius_neighbors', epsilon: 0.1	n_clusters: 10, affinity: 'radius_neighbors', epsilon: 0.2	n_clusters: 10, affinity: 'radius_neighbors', epsilon: 0.2
Ward Clust.	n_clusters: 10, linkage: 'ward'	n_clusters: 10, linkage: 'ward'	n_clusters: 10, linkage: 'ward'	n_clusters: 10, linkage: 'ward'	n_clusters: 10, linkage: 'ward'
Birch	n_clusters: 10, threshold: 0.05, branching_factor: 75	n_clusters: 10, threshold: 0.01, branching_factor: 150	n_clusters: 10, threshold: 0.01, branching_factor: 2	n_clusters: 10, threshold: 0.01, branching_factor: 2	n_clusters: 10, threshold: 0.01, branching_factor: 2
DEC	n_clusters : 10, alpha: 0.1, embedding_size: 2	n_clusters : 10, alpha: 0.1, embedding_size: 5	n_clusters : 10, alpha: 0.1, embedding_size: 10	n_clusters : 10, alpha: 0.1, embedding_size: 2	n_clusters : 10, alpha: 0.1, embedding_size: 2
IDEC	n_clusters : 10, cluster_loss_weight: 0.01, alpha: 0.1, embedding_size: 2	n_clusters : 10, cluster_loss_weight: 1.0, alpha: 0.1, embedding_size: 5	n_clusters : 10, cluster_loss_weight: 1.0, alpha: 0.1, embedding_size: 10	n_clusters : 10, cluster_loss_weight: 1.0, alpha: 0.2, embedding_size: 2	n_clusters : 10, cluster_loss_weight: 1.0, alpha: 1.0, embedding_size: 2
DipDECK	cluster_loss_weight: 0.01, dip_merge_threshold: 0.95, embedding_size: 2, n_clusters_init: 10, max_n_clusters: 15	cluster_loss_weight: 0.01, dip_merge_threshold: 0.95, embedding_size: 2, n_clusters_init: 15, max_n_clusters: 20	cluster_loss_weight: 0.01, dip_merge_threshold: 0.75, embedding_size: 10, n_clusters_init: 10, max_n_clusters: 15	cluster_loss_weight: 0.1, dip_merge_threshold: 0.95, embedding_size: 2, n_clusters_init: 15, max_n_clusters: 20	cluster_loss_weight: 0.1, dip_merge_threshold: 0.75, embedding_size: 2, n_clusters_init: 15, max_n_clusters: 20
DipEncoder	n_clusters: 10, rec._loss_weight: 10.0, embedding_size: 2	n_clusters: 10, rec._loss_weight: 100.0, embedding_size: 5	n_clusters: 10, rec._loss_weight: 10.0, embedding_size: 10	n_clusters: 10, rec._loss_weight: 100.0, embedding_size: 2	n_clusters: 10, rec._loss_weight: 10.0, embedding_size: 2
Mean Shift	cluster_all: True, bandwidth: 0.06	cluster_all: True, bandwidth: 0.06	cluster_all: True, bandwidth: 0.15	cluster_all: True, bandwidth: 0.17	cluster_all: True, bandwidth: 0.19
DPCA	density_threshold: 0.01, distance_threshold: 0.05, gauss_cutoff: True	density_threshold: 0.01, distance_threshold: 0.15, gauss_cutoff: True	density_threshold: 0.01, distance_threshold: 0.2, gauss_cutoff: True	density_threshold: 0.01, distance_threshold: 0.2, gauss_cutoff: False	density_threshold: 0.01, distance_threshold: 0.2, gauss_cutoff: True
DCF	k: 7, beta: 0.991	k: 10, beta: 0.96	k: 20, beta: 0.98	k: 40, beta: 0.99999	k: 40, beta: 0.99999
DDC	radius: 0.06, merge: True	radius: 0.08, merge: True	radius: 0.2, merge: True	radius: 0.22, merge: True	radius: 0.24, merge: True
DBSCAN	epsilon: 0.008, min_samples: 3	epsilon: 0.015, min_samples: 2	epsilon: 0.03, min_samples: 4	epsilon: 0.09, min_samples: 2	epsilon: 0.09, min_samples: 2
OPTICS	cluster_method: 'xi', xi: 0.05, min_cluster_size: 200, predecessor_corr.: True	cluster_method: 'xi', xi: 0.04, min_cluster_size: 150, predecessor_corr.: True	cluster_method: 'xi', xi: 0.15, min_cluster_size: 50, predecessor_corr.: True	cluster_method: 'xi', xi: 0.08, min_cluster_size: 100, predecessor_corr.: True	cluster_method: 'xi', xi: 0.3, min_cluster_size: 10, predecessor_corr.: True
HDBSCAN	min_cluster_size: 6	min_cluster_size: 5	min_cluster_size: 5	min_cluster_size: 6	min_cluster_size: 2
MD-DBSCAN	var: 2.5, k: 6, min_cluster_size: 2	var: 1.8, k: 10, min_cluster_size: 2	var: 1.5, k: 15, min_cluster_size: 2	var: 1.5, k: 40, min_cluster_size: 15	var: 1.2, k: 40, min_cluster_size: 2

Table 3: Clustering results for different dimensionalities for the "compact"-Setting

dimensionality	2-dim		5-dim		10-dim		50-dim		100-dim	
Metric	ARI	NMI	ARI	NMI	ARI	NMI	ARI	NMI	ARI	NMI
k-Means	0.77 ±0.01	0.78 ±0.01	0.73 ±0.05	0.80 ±0.02	0.72 ±0.01	0.80 ±0.01	0.76 ±0.04	0.85 ±0.02	0.66 ±0.06	0.83 ±0.04
LDA-Km	0.78 ±0.00	0.79 ±0.00	0.72 ±0.00	0.81 ±0.00	0.70 ±0.00	0.80 ±0.00	0.74 ±0.00	0.84 ±0.00	0.63 ±0.00	0.81 ±0.00
X-Means	0.84 ±0.00	0.83 ±0.00	0.86 ±0.03	0.86 ±0.01	0.81 ±0.04	0.85 ±0.01	0.94 ±0.04	0.96 ±0.02	0.90 ±0.06	0.95 ±0.03
Spectral Clust.	0.78 ±0.00	0.84 ±0.00	0.67 ±0.00	0.85 ±0.00	0.65 ±0.00	0.80 ±0.00	0.77 ±0.00	0.86 ±0.00	0.75 ±0.00	0.88 ±0.00
SCAR	0.76 ±0.00	0.81 ±0.00	0.80 ±0.00	0.83 ±0.00	0.77 ±0.00	0.80 ±0.00	0.70 ±0.00	0.81 ±0.00	0.67 ±0.03	0.77 ±0.02
SpectACl	0.91 ±0.03	0.90 ±0.01	0.88 ±0.04	0.89 ±0.02	0.91 ±0.00	0.95 ±0.00	0.99 ±0.00	1.00 ±0.00	1.00 ±0.00	1.00 ±0.00
Ward Clust.	0.81	0.83	0.73	0.82	0.74	0.83	0.74	0.85	0.63	0.81
Birch	0.71	0.78	0.43	0.71	0.75	0.84	0.74	0.85	0.63	0.81
DEC	0.49 ±0.03	0.63 ±0.03	0.63 ±0.06	0.76 ±0.04	0.87 ±0.03	0.92 ±0.01	0.48 ±0.11	0.68 ±0.09	0.51 ±0.03	0.73 ±0.03
IDEC	0.69 ±0.02	0.72 ±0.02	0.63 ±0.07	0.78 ±0.03	0.86 ±0.04	0.91 ±0.01	0.75 ±0.08	0.83 ±0.04	0.75 ±0.07	0.88 ±0.03
DipDECK	0.74 ±0.05	0.75 ±0.02	0.72 ±0.07	0.79 ±0.04	0.73 ±0.09	0.8 ±0.03	0.81 ±0.07	0.88 ±0.03	0.75 ±0.03	0.87 ±0.01
DipEncoder	0.73 ±0.05	0.76 ±0.02	0.66 ±0.08	0.77 ±0.04	0.81 ±0.06	0.86 ±0.02	0.73 ±0.10	0.85 ±0.03	0.69 ±0.06	0.83 ±0.03
Mean Shift	0.88	0.83	0.89	0.83	0.91	0.84	0.95	0.87	0.95	0.88
DPCA	0.80	0.80	0.39	0.68	0.78	0.80	0.84	0.86	0.87	0.91
DCF	0.93	0.92	1.00	0.99	0.99	0.99	0.96	0.92	0.95	0.88
DDC	0.82	0.78	0.73	0.77	0.85	0.80	0.93	0.86	0.92	0.83
DBSCAN	0.97	0.95	0.90	0.90	0.90	0.94	0.99	1.00	1.00	1.00
OPTICS	0.98	0.97	0.94	0.97	0.85	0.93	1.00	1.00	1.00	1.00
HDBSCAN	0.95	0.94	0.98	0.97	0.92	0.92	0.99	1.00	1.00	1.00
MD-DBSCAN	0.97	0.96	1.00	1.00	1.00	1.00	0.99	0.99	1.00	1.00

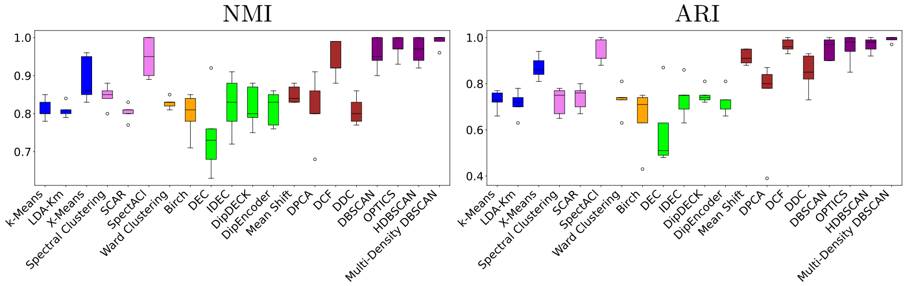

Fig. 5: NMI (left) and ARI (right) across all examined dimensionalities for the setting with "compact" intrinsic dimensionality. The coloration indicates the algorithm category as described in Appendix A.2.

Table 4: Best performing parameters (out of the evaluated ones) for different dimensionalities for the "compact"-Setting

dimensionality	2-dim	5-dim	10-dim	50-dim	100-dim
k-Means	n_clusters: 10, n_init: 3	n_clusters: 10, n_init: 2	n_clusters: 10, n_init: 3	n_clusters: 10, n_init: 7	n_clusters: 10, n_init: 10
LDA-Km	n_clusters: 10, n_init: 7	n_clusters: 10, n_init: 8	n_clusters: 10, n_init: 1	n_clusters: 10, n_init: 1	n_clusters: 10, n_init: 1
X-Means	n_clusters_init: 2, max_n_clusters: 14, n_split_trials: 3	n_clusters_init: 11, max_n_clusters: 12, n_split_trials: 10	n_clusters_init: 6, max_n_clusters: 10, n_split_trials: 10	n_clusters_init: 6, max_n_clusters: 10, n_split_trials: 2	n_clusters_init: 3, max_n_clusters: 10, n_split_trials: 3
Spectral Clust.	n_clusters: 10, affinity: 'nearest_neighbors', nearest_neighbors: 19	n_clusters: 10, affinity: 'nearest_neighbors', nearest_neighbors: 30	n_clusters: 10, affinity: 'nearest_neighbors', nearest_neighbors: 190	n_clusters: 10, affinity: 'nearest_neighbors', nearest_neighbors: 100	n_clusters: 10, affinity: 'nearest_neighbors', nearest_neighbors: 200
SCAR	k: 10, nn: 75, alpha: 0.8, theta: 5000 weighted: True, laplacian: 0, normalize: True	k: 10, nn: 75, alpha: 0.9, theta: 1000 weighted: True, laplacian: 0, normalize: True	k: 10, nn: 30, alpha: 0.6, theta: 4000 weighted: True, laplacian: 0, normalize: True	k: 10, nn: 50, alpha: 0.9, theta: 4000 weighted: True, laplacian: 0, normalize: True	k: 10, nn: 100, alpha: 0.8, theta: 1000 weighted: True, laplacian: 0, normalize: True
SpectACl	n_clusters: 10, affinity: 'radius_neighbors', epsilon: 0.08	n_clusters: 10, affinity: 'radius_neighbors', epsilon: 0.25	n_clusters: 10, affinity: 'radius_neighbors', epsilon: 0.12	n_clusters: 10, affinity: 'radius_neighbors', epsilon: 0.15	n_clusters: 10, affinity: 'radius_neighbors', epsilon: 0.15
Ward Clust.	n_clusters: 10, linkage: 'ward'	n_clusters: 10, linkage: 'ward'	n_clusters: 10, linkage: 'ward'	n_clusters: 10, linkage: 'ward'	n_clusters: 10, linkage: 'ward'
Birch	n_clusters: 10, threshold: 0.01, branching_factor: 2	n_clusters: 10, threshold: 0.01, branching_factor: 2	n_clusters: 10, threshold: 0.01, branching_factor: 2	n_clusters: 10, threshold: 0.01, branching_factor: 2	n_clusters: 10, threshold: 0.01, branching_factor: 2
DEC	n_clusters : 10, alpha: 0.2, embedding_size: 2	n_clusters : 10, alpha: 0.2, embedding_size: 2	n_clusters : 10, alpha: 0.1, embedding_size: 10	n_clusters : 10, alpha: 1.0, embedding_size: 5	n_clusters : 10, alpha: 0.1, embedding_size: 5
IDEC	n_clusters : 10, cluster_loss_weight: 0.01, alpha: 0.7, embedding_size: 2	n_clusters : 10, cluster_loss_weight: 1.0, alpha: 0.1, embedding_size: 5	n_clusters : 10, cluster_loss_weight: 0.1, alpha: 0.1, embedding_size: 10	n_clusters : 10, cluster_loss_weight: 1.0, alpha: 0.7, embedding_size: 10	n_clusters : 10, cluster_loss_weight: 1.0, alpha: 0.4, embedding_size: 2
DipDECK	cluster_loss_weight: 0.01, dip_merge_threshold: 0.85, embedding_size: 2, n_clusters_init: 15, max_n_clusters: 20	cluster_loss_weight: 0.01, dip_merge_threshold: 0.95, embedding_size: 5, n_clusters_init: 15, max_n_clusters: 20	cluster_loss_weight: 0.01, dip_merge_threshold: 0.75, embedding_size: 10, n_clusters_init: 15, max_n_clusters: 20	cluster_loss_weight: 1.0, dip_merge_threshold: 0.95, embedding_size: 2, n_clusters_init: 15, max_n_clusters: 20	cluster_loss_weight: 0.1, dip_merge_threshold: 0.95, embedding_size: 2, n_clusters_init: 15, max_n_clusters: 20
DipEncoder	n_clusters: 10, rec._loss_weight: 10.0, embedding_size: 2	n_clusters: 10, rec._loss_weight: 10.0, embedding_size: 5	n_clusters: 10, rec._loss_weight: 10.0, embedding_size: 10	n_clusters: 10, rec._loss_weight: 10.0, embedding_size: 2	n_clusters: 10, rec._loss_weight: 1000.0, embedding_size: 2
Mean Shift	cluster_all: True, bandwidth: 0.06	cluster_all: True, bandwidth: 0.08	cluster_all: True, bandwidth: 0.11	cluster_all: True, bandwidth: 0.14	cluster_all: True, bandwidth: 0.14
DPCA	density_threshold: 0.01, distance_threshold: 0.07, gauss_cutoff: False	density_threshold: 0.01, distance_threshold: 0.09, gauss_cutoff: False	density_threshold: 0.01, distance_threshold: 0.1, gauss_cutoff: False	density_threshold: 0.01, distance_threshold: 0.15, gauss_cutoff: True	density_threshold: 0.01, distance_threshold: 0.15, gauss_cutoff: True
DCF	k: 8, beta: 0.97	k: 10, beta: 0.95	k: 12, beta: 0.97	k: 50, beta: 0.99999	k: 35, beta: 0.99999
DDC	radius: 0.1, merge: False	radius: 0.1, merge: True	radius: 0.15, merge: True	radius: 0.18, merge: True	radius: 0.19, merge: True
DBSCAN	epsilon: 0.015, min_samples: 3	epsilon: 0.03, min_samples: 2	epsilon: 0.05, min_samples: 8	epsilon: 0.11, min_samples: 2	epsilon: 0.11, min_samples: 2
OPTICS	cluster_method: 'xi', xi: 0.08, min_cluster_size: 150, predecessor_corr.: True	cluster_method: 'xi', xi: 0.03, min_cluster_size: 100, predecessor_corr.: True	cluster_method: 'xi', xi: 0.02, min_cluster_size: 100, predecessor_corr.: True	cluster_method: 'xi', xi: 0.08, min_cluster_size: 10, predecessor_corr.: True	cluster_method: 'xi', xi: 0.15, min_cluster_size: 10, predecessor_corr.: True
HDBSCAN	min_cluster_size: 12	min_cluster_size: 15	min_cluster_size: 3	min_cluster_size: 3	min_cluster_size: 2
MD-DBSCAN	var: 3.0, k: 10, min_cluster_size: 2	var: 1.8, k: 10, min_cluster_size: 2	var: 1.5, k: 10, min_cluster_size: 2	var: 1.2, k: 25, min_cluster_size: 5	var: 1.2, k: 30, min_cluster_size: 15

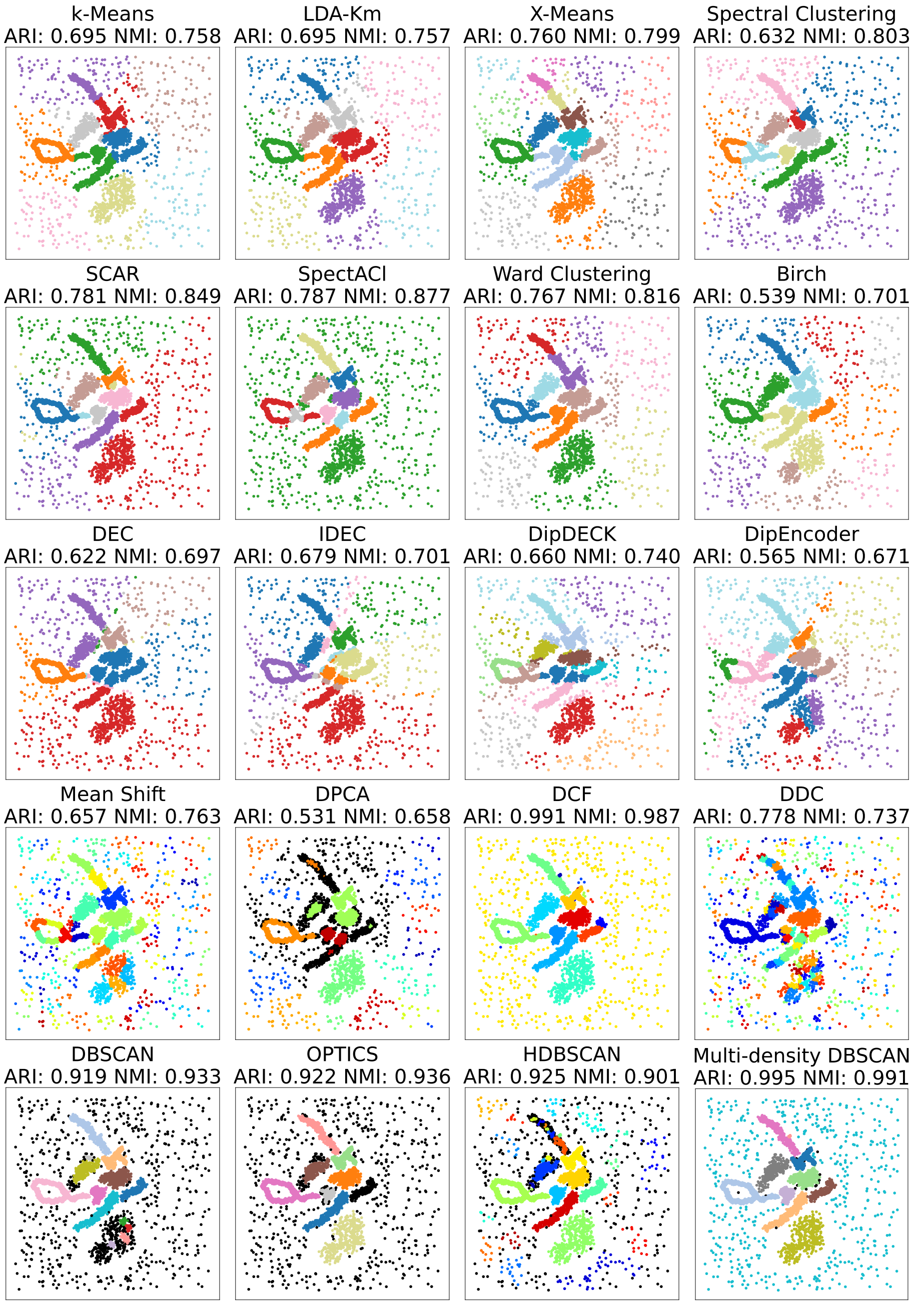

Fig. 6: State-of-the-art and fundamental clustering algorithms on the "mix"-setting dataset generated by DENSIRED with their corresponding performance

Table 5: Best performing parameters (out of the evaluated ones) for the "mix"-Setting

dimensionality	2-dim
k-Means	n_clusters: 10, n_init: 2
LDA-Km	n_clusters: 10, n_init: 1
X-Means	n_clusters_init: 7, max_n_clusters: 13, n_split_trials: 8
Spectral Clust.	n_clusters: 10, affinity: ’nearest_neighbors’, nearest_neighbors: 130
SCAR	k: 10, nn: 15, alpha: 0.9, theta: 1000 weighted: True, laplacian: 0, normalize: True
SpectACl	n_clusters: 10, affinity: ’radius_neighbors’, epsilon: 0.05
Ward Clust.	n_clusters: 10, linkage: ’ward’
Birch	n_clusters: 10, threshold: 0.025, branching_factor: 200
DEC	n_clusters : 10, alpha: 0.1, embedding_size: 2
IDEC	n_clusters : 10, cluster_loss_weight: 0.01, alpha: 0.3, embedding_size: 2
DipDECK	cluster_loss_weight: 0.01, dip_merge_threshold: 0.75, embedding_size: 2, n_clusters_init: 15, max_n_clusters: 20
DipEncoder	n_clusters: 10, reconstruction_loss_weight: 1000.0, embedding_size: 2
Mean Shift	cluster_all: True, bandwidth: 0.05
DPCA	density_threshold: 0.01, distance_threshold: 0.1, gauss_cutoff: True
DCF	k: 9, beta: 0.9
DDC	radius: 0.1, merge: False
DBSCAN	epsilon: 0.015, min_samples: 15
OPTICS	cluster_method: ’xi’, xi: 0.05, min_cluster_size: 250, predecessor_correction: True
HDBSCAN	min_cluster_size: 5
MD-DBSCAN	var: 2.75, k: 10, min_cluster_size: 5

References

1. A. Zimmermann, “Method evaluation, parameterization, and result validation in unsupervised data mining: A critical survey,” *WIREs Data Mining Knowl. Discov.*, vol. 10, no. 2, 2020.
2. H. Demirtas, “A method for multivariate ordinal data generation given marginal distributions and correlations,” *Journal of Statistical Computation and Simulation*, vol. 76, no. 11, pp. 1017–1025, 2006.
3. A. Beer, N. S. Schüler, and T. Seidl, “A generator for subspace clusters,” in *LWDA*, ser. CEUR Workshop Proceedings, vol. 2454. CEUR-WS.org, 2019, pp. 69–73.
4. A. Dries, “Declarative data generation with problog,” in *SoICT*. ACM, 2015, pp. 17–24.
5. J. Dahmen and D. Cook, “SynSys: a synthetic data generation system for healthcare applications,” *Sensors*, vol. 19, no. 5, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/5/1181
6. A. Yale, S. Dash, R. Dutta, I. Guyon, A. Pavao, and K. P. Bennett, “Generation and evaluation of privacy preserving synthetic health data,” *Neurocomputing*, vol. 416, pp. 244–255, Nov. 2020. [Online]. Available: https://doi.org/10.1016/j.neucom.2019.12.136
7. C. M. M. Frey, A. Jungwirth, M. Frey, and R. Kolisch, “The vehicle routing problem with time windows and flexible delivery locations,” *European Journal of Operational Research*, vol. 308, no. 3, pp. 1142–1159, 2023.
8. C. M. M. Frey, A. Züfle, T. Emrich, and M. Renz, “Efficient information flow maximization in probabilistic graphs,” *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 5, pp. 880–894, 2018.
9. D. Libes, D. Lechevalier, and S. Jain, “Issues in synthetic data generation for advanced manufacturing,” in *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 2017, pp. 1746–1754.
10. K. Kailing, H.-P. Kriegel, and P. Kröger, “Density-connected subspace clustering for high-dimensional data,” in *Proceedings of the 2004 SIAM international conference on data mining*. SIAM, 2004, pp. 246–256.
11. Y. Pei and O. R. Zaiane, “A synthetic data generator for clustering and outlier analysis,” 2006.
12. G. W. Milligan, “An algorithm for generating artificial test clusters,” *Psychometrika*, vol. 50, pp. 123–127, 1985.
13. G. W. Milligan and M. C. Cooper, “A study of standardization of variables in cluster analysis,” *J. Classif.*, vol. 5, pp. 181–204, 1988.
14. W. Qiu and H. Joe, “Generation of random clusters with specified degree of separation,” *J. Classif.*, vol. 23, no. 2, pp. 315–334, 2006.
15. V. Melnykov, W.-C. Chen, and R. Maitra, “MixSim: an r package for simulating data to study performance of clustering algorithms,” *Journal of Statistical Software*, vol. 51, pp. 1–25, 2012.
16. J. A. dos Santos, S. T. Iqbal, M. C. Naldi, R. J. G. B. Campello, and J. Sander, “Hierarchical density-based clustering using MapReduce,” *IEEE Trans. Big Data*, vol. 7, no. 1, pp. 102–114, 2021.
17. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, 2011.

18. A. Bryant and K. J. Cios, "RNN-DBSCAN: A density-based clustering algorithm using reverse nearest neighbor density estimates," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 6, pp. 1109–1121, 2018.
19. N. Fachada and D. de Andrade, "Generating multidimensional clusters with support lines," *Knowl. Based Syst.*, vol. 277, p. 110836, 2023.
20. N. Fachada and A. C. Rosa, "generatedata - A 2d data generator," *Softw. Impacts*, vol. 4, p. 100017, 2020.
21. D. L. Steinley and R. Henson, "OCLUS: an analytic method for generating clusters with known overlap," *J. Classif.*, vol. 22, no. 2, pp. 221–250, 2005.
22. C. Shand, R. Allmendinger, J. Handl, A. M. Webb, and J. Keane, "HAWKS: evolving challenging benchmark sets for cluster analysis," *IEEE Trans. Evol. Comput.*, vol. 26, no. 6, pp. 1206–1220, 2022.
23. F. Iglesias, T. Zseby, D. C. Ferreira, and A. Zimek, "MDCGen: Multidimensional dataset generator for clustering," *J. Classif.*, vol. 36, no. 3, pp. 599–618, 2019.
24. J. R. Vennam and S. Vadapalli, "SynDECA: A tool to generate synthetic datasets for evaluation of clustering algorithms," in *COMAD*. Computer Society of India, 2005, pp. 27–36.
25. D. A. Rachkovskij and E. M. Kussul, "DataGen: a generator of datasets for evaluation of classification algorithms," *Pattern Recognit. Lett.*, vol. 19, no. 7, pp. 537–544, 1998.
26. J. Gan and Y. Tao, "DBSCAN revisited: Mis-claim, un-fixability, and approximation," in *SIGMOD Conference*. ACM, 2015, pp. 519–530.
27. ——, "On the hardness and approximation of euclidean DBSCAN," *ACM Trans. Database Syst.*, vol. 42, no. 3, pp. 14:1–14:45, 2017.
28. X. Huang, T. Ma, C. Liu, and S. Liu, "GriT-DBSCAN: A spatial clustering algorithm for very large databases," *Pattern Recognit.*, vol. 142, p. 109658, 2023.
29. A. Pape and D. Dubin, "Clusutils, sourceforge," 2000, accessed on Mar 22, 2024. [Online]. Available: https://clusutils.sourceforge.net/
30. P. M. Murphy, "Uci repository of machine learning databases," *http://www. ics. uci. edu/˜ mlearn/MLRepository. html*, 1994.
31. S. P. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. 28, no. 2, pp. 129–136, 1982.
32. C. H. Q. Ding and T. Li, "Adaptive dimension reduction using discriminant analysis and K-means clustering," in *ICML*, ser. ACM International Conference Proceeding Series, vol. 227. ACM, 2007, pp. 521–528.
33. T. Hastie, J. H. Friedman, and R. Tibshirani, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, ser. Springer Series in Statistics. Springer, 2001.
34. D. Pelleg and A. W. Moore, "X-means: Extending k-means with efficient estimation of the number of clusters," in *ICML*. Morgan Kaufmann, 2000, pp. 727–734.
35. T. Ishioka, "An expansion of x-means for automatically determining the optimal number of clusters aˆeur" progressive iterations of k-means and merging of the clusters," in *Computational Intelligence*. IASTED/ACTA Press, 2005, pp. 91–96.
36. U. von Luxburg, "A tutorial on spectral clustering," *Stat. Comput.*, vol. 17, no. 4, pp. 395–416, 2007.
37. S. X. Yu and J. Shi, "Multiclass spectral clustering," in *ICCV*. IEEE Computer Society, 2003, pp. 313–319.
38. J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, 2000.

39. E. Hohma, C. M. M. Frey, A. Beer, and T. Seidl, “SCAR - spectral clustering accelerated and robustified,” *Proc. VLDB Endow.*, vol. 15, no. 11, pp. 3031–3044, 2022.
40. A. Bojchevski, Y. Matkovic, and S. Günnemann, “Robust spectral clustering for noisy data: Modeling sparse corruptions improves latent embeddings,” in *KDD.* ACM, 2017, pp. 737–746.
41. C. C. Fowlkes, S. J. Belongie, F. R. K. Chung, and J. Malik, “Spectral grouping using the nyström method,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 2, pp. 214–225, 2004.
42. S. Hess, W. Duivesteijn, P. Honysz, and K. Morik, “The SpectACl of nonconvex clustering: A spectral approach to density-based clustering,” in *AAAI.* AAAI Press, 2019, pp. 3788–3795.
43. J. H. Ward Jr, “Hierarchical grouping to optimize an objective function,” *Journal of the American statistical association*, vol. 58, no. 301, pp. 236–244, 1963.
44. D. Wishart, “256. note: An algorithm for hierarchical classifications,” *Biometrics*, pp. 165–170, 1969.
45. S. C. Johnson, “Hierarchical clustering schemes,” *Psychometrika*, vol. 32, no. 3, pp. 241–254, 1967.
46. R. K. Blashfield, “Mixture model tests of cluster analysis: accuracy of four agglomerative hierarchical methods.” *Psychological Bulletin*, vol. 83, no. 3, p. 377, 1976.
47. T. Zhang, R. Ramakrishnan, and M. Livny, “BIRCH: an efficient data clustering method for very large databases,” in *SIGMOD Conference.* ACM Press, 1996, pp. 103–114.
48. J. Xie, R. B. Girshick, and A. Farhadi, “Unsupervised deep embedding for clustering analysis,” in *ICML*, ser. JMLR Workshop and Conference Proceedings, vol. 48. JMLR.org, 2016, pp. 478–487.
49. X. Guo, L. Gao, X. Liu, and J. Yin, “Improved deep embedded clustering with local structure preservation,” in *IJCAI.* ijcai.org, 2017, pp. 1753–1759.
50. C. Leiber, L. G. M. Bauer, B. Schelling, C. Böhm, and C. Plant, “Dip-based deep embedded clustering with k-estimation,” in *KDD.* ACM, 2021, pp. 903–913.
51. J. A. Hartigan and P. M. Hartigan, “The dip test of unimodality,” *The annals of Statistics*, pp. 70–84, 1985.
52. C. Leiber, L. G. M. Bauer, M. Neumayr, C. Plant, and C. Böhm, “The dipencoder: Enforcing multimodality in autoencoders,” in *KDD.* ACM, 2022, pp. 846–856.
53. D. Comaniciu and P. Meer, “Mean Shift: A robust approach toward feature space analysis,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 603–619, 2002.
54. K. Fukunaga and L. D. Hostetler, “The estimation of the gradient of a density function, with applications in pattern recognition,” *IEEE Trans. Inf. Theory*, vol. 21, no. 1, pp. 32–40, 1975.
55. Y. Cheng, “Mean shift, mode seeking, and clustering,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 8, pp. 790–799, 1995.
56. A. Rodriguez and A. Laio, “Clustering by fast search and find of density peaks,” *science*, vol. 344, no. 6191, pp. 1492–1496, 2014.
57. J. Tobin and M. Zhang, “DCF: an efficient and robust density-based clustering method,” in *ICDM.* IEEE, 2021, pp. 629–638.
58. H. Jiang and S. Kpotufe, “Modal-set estimation with an application to clustering,” in *AISTATS*, ser. Proceedings of Machine Learning Research, vol. 54. PMLR, 2017, pp. 1197–1206.

59. R. Hyde and P. Angelov, "Data density based clustering," in *UKCI*. IEEE, 2014, pp. 1–7.
60. M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *KDD*. AAAI Press, 1996, pp. 226–231.
61. E. Schubert, J. Sander, M. Ester, H. Kriegel, and X. Xu, "DBSCAN revisited, revisited: Why and how you should (still) use DBSCAN," *ACM Trans. Database Syst.*, vol. 42, no. 3, pp. 19:1–19:21, 2017.
62. M. Ankerst, M. M. Breunig, H. Kriegel, and J. Sander, "OPTICS: ordering points to identify the clustering structure," pp. 49–60, 1999.
63. E. Schubert and M. Gertz, "Improving the cluster structure extracted from OPTICS plots," in *LWDA*, ser. CEUR Workshop Proceedings, vol. 2191. CEUR-WS.org, 2018, pp. 318–329.
64. R. J. G. B. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in *PAKDD (2)*, ser. Lecture Notes in Computer Science, vol. 7819. Springer, 2013, pp. 160–172.
65. L. McInnes and J. Healy, "Accelerated hierarchical density based clustering," in *ICDM Workshops*. IEEE Computer Society, 2017, pp. 33–42.
66. W. M. Ashour and S. Sunoallah, "Multi density DBSCAN," in *IDEAL*, ser. Lecture Notes in Computer Science, vol. 6936. Springer, 2011, pp. 446–453.
67. S. T. Mai, I. Assent, and M. Storgaard, "AnyDBC: An efficient anytime density-based clustering algorithm for very large complex datasets," in *KDD*. ACM, 2016, pp. 1025–1034.
68. T. Ullmann, A. Beer, M. Hünemörder, T. Seidl, and A. Boulesteix, "Over-optimistic evaluation and reporting of novel cluster algorithms: an illustrative study," *Adv. Data Anal. Classif.*, vol. 17, no. 1, pp. 211–238, 2023.
69. L. Hubert and P. Arabie, "Comparing partitions," *J. Classif.*, vol. 2, pp. 193–218, 1985.
70. A. Strehl and J. Ghosh, "Cluster ensembles — A knowledge reuse framework for combining multiple partitions," *J. Mach. Learn. Res.*, vol. 3, pp. 583–617, 2002.
71. C. Leiber, L. Miklautz, C. Plant, and C. Böhm, "Benchmarking deep clustering algorithms with clustpy," in *ICDM (Workshops)*. IEEE, 2023, pp. 625–632.