Density-based Clustering by Means of Bridge Point Identification

Density-based Clustering by Means of Bridge Point Identification

Luca Colomba, Luca Cagliero, *Member, IEEE,* and Paolo Garza, *Member, IEEE*

Abstract— Density-based clustering focuses on defining clusters consisting of contiguous regions characterized by similar densities of points. Traditional approaches identify core points first, whereas more recent ones initially identify the cluster borders and then propagate cluster labels within the delimited regions. Both strategies encounter issues in presence of multi-density regions or when clusters are characterized by noisy borders. To overcome the above issues, we present a new clustering algorithm that relies on the concept of bridge point. A bridge point is a point whose neighborhood includes points of different clusters. The key idea is to use bridge points, rather than border points, to partition points into clusters. We have proved that a correct bridge point identification yields a cluster separation consistent with the expectation. To correctly identify bridge points in absence of a priori cluster information we leverage an established unsupervised outlier detection algorithm. Specifically, we empirically show that, in most cases, the detected outliers are actually a superset of the bridge point set. Therefore, to define clusters we spread cluster labels like a wildfire until an outlier, acting as a candidate bridge point, is reached. The proposed algorithm performs statistically better than state-of-the-art methods on a large set of benchmark datasets and is particularly robust to the presence of intra-cluster multiple densities and noisy borders.

Index Terms—Density-based Clustering, Outlier detection, Unsupervised learning.

1 Introduction

Clustering is among the most popular unsupervised data mining tasks. It addresses the definition of groups of data points, where each group contains highly similar points whereas points in separate groups are dissimilar [1]. Density-based methods encompass a large body of clustering algorithms focused on analyzing the neighborhood of each data point. The pioneering DBSCAN algorithm [2] analyzes the density of points within the neighborhood of a given point to decide whether the latter can be classified as a *core point*, i.e., a point with a high density of points in the surrounding region, and used as starting point for defining a new cluster. Although several DBSCAN variants have been proposed in literature (e.g., OPTICS [3], HDBSCAN [4]), they show limited performances in presence of multi-density regions, i.e., clusters consisting of a variety of regions with different densities of points [5]. In particular, the presence of variable densities of points within the same cluster (i.e., the intra-cluster multiple density) is known to be particularly challenging.

Recently, the data mining community has explored alternative density-based strategies aimed at identifying the points lying in the cluster border first, namely the border points [6], [7]. A correct detection of the border points has shown to be effective in identifying inter-cluster multiple-density regions. However, the performances can be suboptimal while coping with either intra-cluster multiple densities or clusters with noisy borders. The main reason is that the clustering algorithm likely overestimate the number of border points thus yielding an excessive cluster fragmentation.

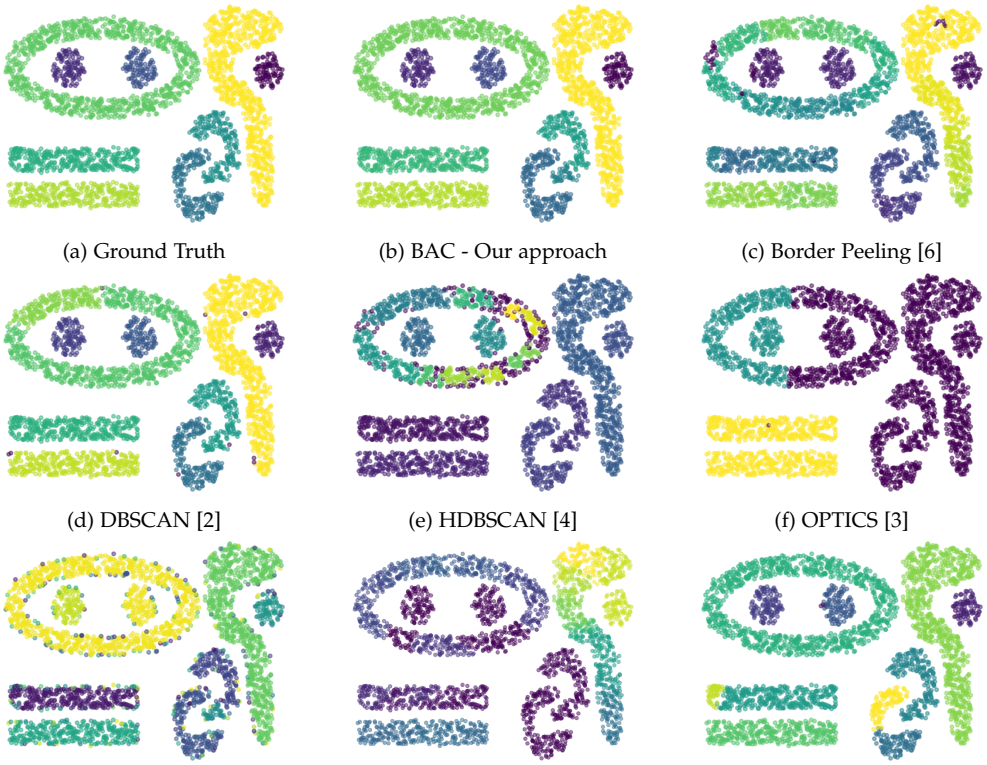
The present work focuses on addressing the aforesaid issues by introducing a new type of data points, namely the *bridge point*. A bridge point is a point whose neighborhood includes points of different clusters. They symbolically provide inter-cluster connections, thus acting as *bridges* between separate clusters. The key idea behind this work is to identify bridge points first as they indicate clusters' *frontiers* more explicitly than border points. In Section 4 we will prove that a preliminary bridge point identification yields a cluster separation that is consistent with the expected clustering outcome (i.e., the generated clusters do not include points with different cluster labels in the ground truth).

In an unsupervised context, bridge points are a apriori unknown. Hence, we leverage outlier detection methods to discover candidate bridge points. Specifically, since points lying in cluster frontiers are likely to be detected by outlier detection methods we propose a clustering algorithm that performs outlier detection first to identify candidate bridge points, and then assigns cluster labels to data points accordingly. The underlying hypothesis that bridge points are a subset of the outlier set is confirmed by empirical evidence achieved on a large set of synthetic and real-world benchmark datasets.

We spread cluster labels within regions delimited by candidate bridge points. To prevent excessive cluster fragmentation, label propagation is stopped only when all points in its local neighborhood are outliers (i.e., candidate bridge points). In such a way, the clustering algorithm achieves a robustness to intra-cluster multiple densities and noisy borders superior to existing approaches. The performance comparisons with state-of-the-art algorithms were conducted on 35 synthetic and 10 real-world benchmark datasets [11] included in the ClueMiner project [12].

• *L. Colomba, L. Cagliero, and P. Garza are with the Dipartimento di Automatica e Informatica, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129, Torino, Italy.*
E-mail: {luca.colomba,luca.cagliero,paolo.garza}@polito.it

Manuscript received ...



(a) Ground Truth (b) BAC - Our approach (c) Border Peeling [6]

(d) DBSCAN [2] (e) HDBSCAN [4] (f) OPTICS [3]

(g) AUTOCLUST [8] (h) DADC [9] (i) DenMune [10]

Fig. 1: Complex9 dataset: comparison between the outcomes of different clustering algorithms.

The main paper contributions can be summarized as follows.

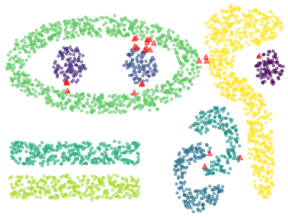
- **A new type of data point**. We introduce the bridge point concept and use it to identify density-based clusters. We also empirically show that outlier detection methods achieve a very high recall in detecting bridge points. Therefore, they are particularly suitable for detecting candidate bridge points in an unsupervised context.
- **New clustering pipeline**. We design and develop a new clustering algorithm that exploits outlier detection methods to identify candidate bridge points first and then uses candidate bridge points to define clusters. Assuming that the complete set of bridge points is given, we also prove the algorithm correctness, i.e., we are able to separate points belonging to different clusters (see Theorem 1).
- **Extensive experimental evaluation.** We compare the performance of the proposed clustering algorithm with both traditional and recent approaches on a large set of synthetic and real-world benchmark datasets. The results show that the proposed approach performs significantly better than state-of-the-art approaches and is particularly robust to the presence of intra-cluster multi-density regions and noisy borders.

The remainder of the paper is organized as follows. Section 2 presents a motivating example. Section 3 reviews the prior work. Section 4 introduces the concept of bridge point and provides a data-driven characterization whereas Section 5 describes how to leverage an outlier detection step to identify candidate bridge points. Sections 6 and 7 respectively present the clustering pipeline relying on candidate bridge points and summarize the empirical results achieved on benchmark datasets characterized by various data distributions. Finally, Section 8 draws conclusions and discusses the future research agenda.

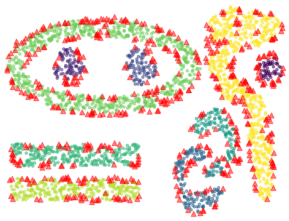
2 Motivating Example

We present here the outcomes of various clustering algorithms on a prime example, i.e., a dataset characterized by two main issues: (i) some regions have different densities and (ii) cluster borders are quite noisy. We will show that the data distribution described above is particularly challenging for existing density-based approaches, including traditional density-based algorithms (e.g., [2], [3], [4]), recently proposed approaches based on border point identification (e.g., Border Peeling [6]), and density-peak clustering (e.g., [9], [10]).

In Figure 1a we show the expected clusters (denoted by *Ground Truth*). They are characterized by heterogeneous shapes, variable densities (particularly, the ring-like cluster in the center), and not well defined borders (particularly, the clusters on the right-hand side). The outcomes of the density-based algorithms, which are depicted in Figures 1d, 1e, and 1f exemplify the difficulties encountered by the respective algorithms to cope with multi-density regions (e.g., the ring-like cluster is split into several parts). Figure 1c depicts the clusters defined by Border Peeling [6]. Unlike DBSCAN and OPTICS, Border Peeling is able to correctly



(a) Bridge points



(b) Outliers

Fig. 2: Our approach: an insight into bridge points and outliers. Complex9 dataset.

separate the clusters with less pronounced borders, but fails to correctly define the ring-like cluster due to the presence of intra-cluster variable density regions. Figure 1h reports the clusters produced by DADC [9], which encounters problems while trying to merge regions with variable density of points or to split clusters with similar densities. Finally, DenMune [10] splits some of the clusters characterized by intra-cluster multiple densities (see Figure 1i).

Our clustering approach (see Figure 1b) produces well separated clusters thanks to the application of outlier detection followed by bridge point identification. To gain insights into the clustering process, Figures 2a and 2b respectively show the red colored bridge points and the detected outliers. For example, the bridge points placed inside and in the right-hand side of the ring-like cluster are useful for separating it from the two nested globular clusters and from the external one. The outlier detection step has been shown to accurately detect a superset of the bridge points, thus yielding an effective cluster separation.

3 Related works

The study and development of clustering techniques have received indisputable interest from the data mining and machine learning communities. Clustering algorithms have been already applied to a wide range of application contexts and data types, including, for instance, geo-spatial data [13], social and multimedia data [14], and Big data collections [15].

According to [16], clustering algorithms can be classified as *partitional*, if they produce unnested clusters where each data point belongs to exactly one cluster, or *hierarchical*, otherwise. An alternative categorization divides the clustering algorithms as *complete* or *partial* [16]. Specifically, in a complete clustering every object is assigned to at least one cluster, whereas in a partial clustering points can remain unassigned. The clustering approach proposed in this paper is partitional and complete.

Among the partitional algorithms, one notable group consists of the density-based approaches. They aim at generating clusters based on the density of points in the regions of the input data space. DBSCAN [2] was the pioneering density-based algorithm. It focuses on first discovering points with a dense neighborhood (i.e., the core points), generate clusters from core points, and then assign the remaining points to the previously defined clusters. The aforesaid approach may encounter problems while dealing with multi-density regions. OPTICS [3] and HDBSCAN [4] tried to alleviate the problem of multi-density clustering by using ad hoc structures, i.e., the reachability plots and hierarchical structures. However, since they do not consider the border of a dense region in the aggregation phase, they tend to label a large number of border points as noise [17].

To specifically cope with datasets characterized by multi-density regions, a relevant research effort has been devoted to detecting density peaks (DPs) (e.g., [18], [19], [20], [21]). As a drawback, the aforesaid algorithms tend to over-fragment the clusters in the presence of intra-cluster regions with variable density. The works presented in [9], [22], [23] focused on overcoming the issue by proposing optimized neighbor search methods, whereas in [20] DPs are detected using decision graphs. Although domain-adaptive density measurements can reveal the presence of multiple density maximums within the same cluster [9], they may encounter issues in defining the cluster backbone.

Alternative approaches (e.g., [10], [24], [25]) proposed to identify clusters' backbones and assign labels based on the presence of density-peak regions. Specifically, the authors in [10] adopted the K-Mutual-Neighbors consistency [26] and a voting mechanism to classify points as strong, weak and noise points. Consequently, cluster backbones are expanded starting from strong points to weak points. Instead, [24] proposed a fuzzy neighborhood kernel to compute local densities, determine cluster centers, and identify cluster backbones. In other works, researchers focused on improving density-peak clustering by providing different definitions of density [25], [27]. Specifically, [25] used only relative density relationships to mitigate the negative effects of inter-cluster multiple-density, meanwhile [27] redefined DP clustering based on a weighted k-Nearest Neighbors distance and on a geodesic-based δ distance. Instead, [28] focused their work on the automated analysis of the decision graph using a gap-based methodology to automatically identify clusters' centers.

A clustering strategy that complements density peak detection methods has been recently presented in [6]. The authors aimed at identifying cluster borders first, rather than density peaks, and then peel the borders until the cluster core is reached. The latter approach has been shown to be particularly effective in handling clusters with different densities, but tends to produce an undesired cluster fragmentation when multi-density regions are included within the same cluster.

Distance graphs can be also exploited to identify connected regions of points. For instance, AUTOCLUST [8] performed clustering by leveraging a graph-based data representation, where edges connecting pairs of points are weighted by the corresponding pairwise distance. The proposed algorithm extracts connected components from a

pruned graph version, which excludes edges corresponding to too close or faraway connections (e.g., connections between points located in different space regions).

Summary of differences with prior works.

- DBSCAN [2],OPTICS [3] and HDBSCAN [4]: they identify core points with a dense neighborhood rather than bridge points representing cluster frontiers.
- Density-peak clustering [9], [10], [18], [19], [20], [21], [24], [25]: our approach prioritizes the identification of points that are likely to lie on cluster borders. In this sense, it is complementary to density-peak clustering strategies.
- Border Peeling [6]: our main goal is not to detect and peel cluster borders, but rather to identify bridge points, which represent *cluster frontiers* (not necessarily lying on the border). The proposed solution focuses on preventing excessive cluster fragmentation. To the best of our knowledge, this work is the first attempt to exploit bridge points and outlier detection to define clusters.
- AUTOCLUST [8]: our purpose is to detect bridge points by means of outlier detection rather than simply considering the connected subgraphs. To avoid producing excessive cluster fragmentation we relax the arbitrary assumption that bridges among cluster consist of short links. Furthermore, we inherently manage the cases when different clusters are associated with multiple bridge paths.
- Semi-supervised clustering (e.g., [29]): this paper addresses the clustering problem in an *unsupervised* context.

4 Bridge point type definition and characterization

This section formalizes the problem addressed in this paper, introduces the new type of data points, and describes their main properties.

Problem statement. Let $\mathcal{D}$=$\{p_1, p_2, \ldots, p_n\}$ be an input dataset, where p_i=(x_1^i,x_2^i, ..., x_d^i)$\in \mathbb{R}^d$ [$1 \leq i \leq n$] will be hereafter denoted as *data point*, whereas x_j^i will be the value of its *j-th dimension* [30].

We address the *hard partitional clustering* problem, which entails finding a q-partition of $\mathcal{D}$, namely $\mathcal{C} = \{C_1, C_2, \ldots, C_q\}$ ($q \leq n$) such that

- $C_j \neq \emptyset$, j=1, ..., q
- $\bigcup_{j=1}^{q} C_j = \mathcal{D}$
- $C_j \cap C_l = \emptyset$ $j, l = 1, \ldots, q$ and $j \neq l$

Without any loss of generality, hereafter we will denote every partition in $\mathcal{C}$ as a *cluster*. The clustering algorithm assigns each data data point $p_i \in \mathcal{D}$ to a cluster C_i, $i \in [1, \ldots, q]$. p_i is labeled as c_i.

Hereafter, unless otherwise specified, we will consider a Euclidean space.

Neighborhood of a data point. Density-based clustering relies on the concept of local neighborhood of data point, which is formalized below.

Definition 1 (k-nearest neighborhood). Let $k \in \mathbb{Z}^+$ be a user-specified positive integer value and let $p_i \in \mathcal{D}$ be an arbitrary data point. The k-nearest neighborhood of p_i, hereafter denoted as $NN_k(p_i)$, is the set of its k nearest points.

For our convenience, we represent the relationship between a data point and its k neighbors using a graph-based model.

Definition 2 (k-nearest neighborhood graph). Let $\mathcal{D}$ be an input dataset and let $\mathbb{G}$ be an undirected graph whose nodes are data points in $\mathcal{D}$ whereas edges link pairs of points p_i, $p_j \in \mathcal{D}$ such that either $p_j \in NN_k(p_i)$ or $p_i \in NN_k(p_j)$.

Bridge point. We now introduce the concept of bridge point. A bridge point is a data point p_i whose 'true' cluster label differs from the true cluster label of at least one of the points in its k-nearest neighborhood. In a nutshell, bridge points represent cluster "frontiers" because in their neighborhoods there are points not belonging to the same clusters of the bridge points. A more formal definition follows.

Definition 3 (Bridge point). Let $\mathcal{D}$ be a dataset, $\mathcal{C}$ be the optimal clustering[1] for $\mathcal{D}$ and c_i be the cluster label assigned in $\mathcal{C}$ to an arbitrary point in $p_i \in \mathcal{D}$. p_i is a bridge point if

$$\exists p_j \in NN_k(p_i) \mid c_i \neq c_j$$

Examples of bridge points, for different values of k, are shown in Figures 3b and 3d. Bridge points are depicted as red triangles. According to Definition 3, the higher k the higher the number of bridge points. The reason is that by increasing the neighborhood size the likelihood of finding points labeled with a different cluster increases as well.

Regardless of the value of k, bridge points indicate the frontier between two clusters. Thus, their identification is deemed as particularly useful for cluster definition, as empirically shown later on.

Bridge-aware clustering. We now refine the concept of neighborhood graph introduced earlier by assuming to have already correctly identified all bridge points in the input dataset.

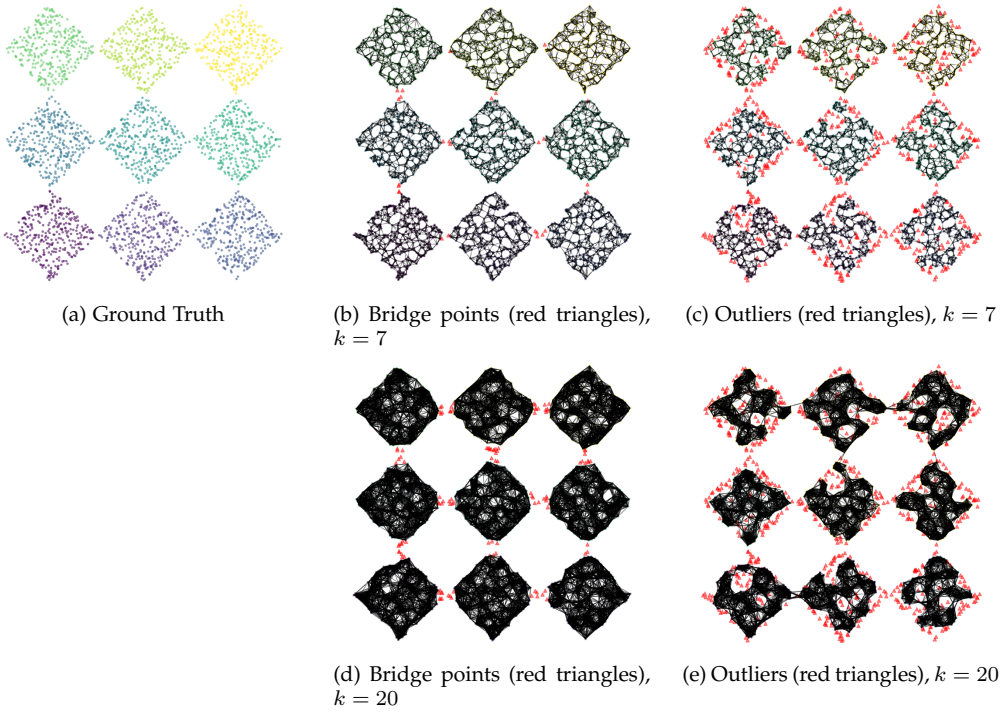
A bridge-aware k-nearest neighborhood graph connects data points to all the points in the local neighborhood of a given data point, except for the bridge points.

Definition 4 (Bridge-aware k-nearest neighborhood graph). Let $\mathcal{D}$ be an input dataset and let $\mathbb{G}$ be a k-nearest neighborhood graph built on $\mathcal{D}$. Let $\mathcal{C}^e$ be the expected clustering outcome for $\mathcal{D}$ and let $\mathcal{B}$ be the corresponding set of bridge points in $\mathcal{D}$. The *bridge-aware* k-nearest neighborhood graph $\mathbb{G}_{ba}$ is the graph derived from $\mathbb{G}$ by pruning all the edges in $\mathbb{G}$ that connect each bridge point in $\mathcal{B}$ to any other point, except for its nearest point in $\mathcal{D}$.

By assigning a different cluster label to each connected component in $\mathbb{G}_{ba}$, we can yield a clustering outcome that preserves cluster separation according to the expected outcome.

Theorem 1 (Clustering based on bridge-aware k-nearest neighborhood graph). Let $\mathcal{D}$ be an input dataset, let $\mathcal{B}$

1. It corresponds to the "expected" outcome.



(a) Ground Truth

(b) Bridge points (red triangles), $k = 7$

(c) Outliers (red triangles), $k = 7$

(d) Bridge points (red triangles), $k = 20$

(e) Outliers (red triangles), $k = 20$

Fig. 3: Examples of bridge points and outliers (candidate bridge points). Diamond9 dataset.

be set of bridge points defined in compliance with the expected clustering outcome $\mathcal{C}^e$, and let $\mathbb{G}_{ba}$ be the corresponding bridge-aware k-nearest neighborhood graph generated on top of $\mathcal{D}$, where each connected component has a distinct cluster label. Any clustering outcome $\mathcal{C}$ such that
(i) each cluster consists of the subset of points corresponding to a connected component in $\mathbb{G}_{ba}$ and
(ii) every bridge point is assigned to the same cluster as the nearest point in its neighborhood
preserves cluster separation given by $\mathcal{C}^e$, i.e., there exist not pairs of points belonging to different clusters in $\mathcal{C}^e$ that have the same cluster label in $\mathcal{C}$.

Proof of Theorem 1. By contradiction. Let (p_r, p_s) be an arbitrary pair of non-bridge data points in $\mathcal{D}$. Let (c^e_r, c^e_s) and (c_r, c_s) be the pairs of the corresponding cluster labels in $\mathcal{C}^e$ and $\mathcal{C}$, respectively. There exists at least one pair (p_r, p_s) satisfying the following properties:
(1) $c^e_r \neq c^e_s$
(2) $c_r = c_s$

According to (1) and in compliance with Definition 4, p_r and p_s do not belong to the same connected component in $\mathbb{G}_{ba}$, whereas according to (2) p_r and p_s must belong to the same connected component in $\mathbb{G}_{ba}$ because, according to Point (i) of Theorem 1, each cluster in $\mathcal{C}$ is a subset of a connected component in $\mathbb{G}_{ba}$. This is a contradiction. ■

Therefore, bridge point identification allows us to achieve a desirable cluster separation. Unfortunately, bridge point classification is a priori unknown in an unsupervised context. In Section 5 we will address the use of outlier detection methods to identify candidate bridge points.

Notice that the notable property of the clustering result stated in Theorem 1 is a *necessary but not sufficient condition* to achieve exactly the expected clustering, because a single cluster in the expected outcome may be further split into more than one clusters according to the clustering generated on the basis of the bridge-aware data representation.

5 Bridge Point Identification Based on Outlier Detection

To identify bridge points we propose to exploit unsupervised outlier detection methods. Outliers are instances of data that deviates significantly from the normal objects [1]. According to the type of considered outlier, normal objects may include either the rest of the data set (in this scenario significant deviations are commonly denoted by *global outliers*) or a selected context (data inconsistencies are here denoted by *contextual outliers*).

Our main goal is to find inconsistencies in data points with respect to their local region. We address this task using either global or contextual outlier detection methods (as long as the concept of context is properly defined). Hence, hereafter we will disregard the case of *collective outliers*, in which a group of data objects deviates significantly from the normal case even if individual objects may not be outliers.

Bridge points belonging to clusters' "frontiers" are likely to be deemed as deviations from the normal objects within the local area. Hence, we explore the relation between bridge points and outliers, empirically showing that the outliers detected by the most established methods (e.g., [31], [32], [33], [34], [35]) are actually a superset of the bridge point set. Then, in Section 6 we will present a cluster labeling procedure that will be applied on top of the outlier detection step to define the output clusters.

In Figure 4 we plot the scatter plot showing the distribution of the analyzed benchmark datasets by varying

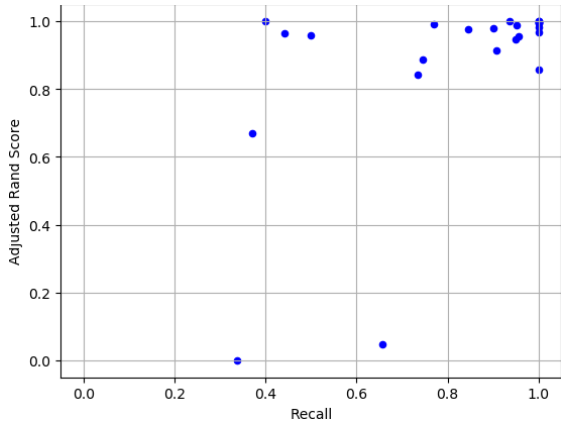


Fig. 4: Dataset distribution. Clustering performance vs. recall of the bridge point identification step based on outlier detection. Performance metric: Adjusted Rand Score. Outlier detection method: LOF [37].

the recall of the bridge point identification step, i.e., the percentage of bridge points that have been identified by the outlier detection step, and the clustering performance achieved by our method (measured by the adjusted rand score index [36]). More details about the clustering algorithm, the analyzed dataset and the used evaluated metrics will be reported in Sections 6 and 7, respectively.

The takeaways from this experiments can be summarized as follows: (i) On ∼70% of the analyzed datasets the outlier detection step was able to recognize over 75% of the bridge points, and (ii) on average, the higher the recall the outlier detection achieves, the better the clustering performance.

Result (i) supports the use of outlier detection methods to perform candidate bridge point identification. Result (ii) strengthens the hypothesis, previously discussed in Section 4, that bridge point identification is particularly relevant to achieve high-quality clustering performance.

5.1 Relation between detected outliers and noise data

Outliers are different from noise data. Outliers' deviation from normal data is commonly due to a different data generation mechanism, whereas noise is random error or variance in a measured variable [1].

We empirically study the relation between detected outliers and noise points on a subset of synthetic datasets in which labeled noise points are injected[2]. The outliers automatically detected by the unsupervised approach averagely cover 83% of the noise points. More detailed results can be found in Appendix B. Since the presence of noise points in the cluster frontiers may bias the process of bridge point identification, as future work we plan to filter out noise using ad hoc strategies prior to/during the outlier detection step (see Section 8).

6 The Bridge-Aware Clustering pipeline

This section presents a new density-based clustering pipeline, namely *Bridge-Aware Clustering* (BAC), which is focused on the identification of cluster frontiers consisting of bridge points.

BAC, whose pipeline is depicted in Figure 5, consists of three main steps:

1) *Candidate bridge point identification* (see Section 6.1).
2) *Bridge-aware k-nearest neighborhood graph creation* (see Section 6.2).
3) *Cluster detection and labeling* (see Section 6.3).

Step (1) leverages the outcomes of an outlier detection step to identify candidate bridge points. Step (2) creates the bridge-aware k-nearest neighborhood graph $\mathbb{G}_{ba}$, according to Definition 4. It relies on the assumptions, previously discussed in Section 5, that the outliers detected at Step (1) include, in the majority of the datasets, the actual bridge points. Finally, Step (3) clusters data points by assigning a different cluster label to each connected component in $\mathbb{G}_{ba}$.

6.1 Candidate bridge point identification

This section addresses the identification of the bridge points, which represent the clusters' frontiers (e.g., the red triangles in Figure 5). As discussed in Section 5, the aforesaid task is conveniently accomplished by means of outlier detection. Outliers have shown to be, to a good approximation, a superset of the bridge point set including also border and noise points. For the sake of clarity, hereafter we will denote the outcome of the outlier detection step as the *candidate* bridge points.

In principle, any unsupervised outlier detection method can be integrated in BAC. Clearly, the choice is influenced by the underlying data characteristics, in terms of, for instance, dimensionality, complexity, and domains. These features could make a specific outlier detection method more appropriate than others. At this stage, the availability of domain-specific knowledge is a clear advantage.

In Section 7 we explore the effect of using different algorithms and parameter settings. Based on the results achieved on a large set of benchmark datasets, Local Outlier Factor (LOF) [37] turned out to be the recommended strategy to identify candidate bridge points (unless otherwise specified). LOF compares the local density of a point with those of its neighbors and labels as outlier those points that have a substantially lower density than their neighbors.

Figures 3c and 3e show the outliers (depicted as red triangles) identified by LOF on a representative dataset by setting two different k values. From the comparison between the detected outliers and the actual bridge points (depicted in Figures 3b and 3d) it turned out that LOF was actually able to detect a superset of the bridge points mixed with some border points.

6.2 Bridge-aware k-nearest neighborhood graph creation

Given the input dataset $\mathcal{D}$ and the set of candidate bridge points returned by the previous step, we first create the k-nearest neighborhood graph $\mathbb{G}$ and then the bridge-aware k-nearest neighborhood graph $\mathbb{G}_{ba}$ in compliance with Definitions 2 and 4. The creation of the k-nearest neighborhood graph is a three-step process. (1) For each point we identify its k-nearest neighbors. (2) We build a graph $\mathbb{G}$ where each

2. cluto-t4-8k, cluto-t2-4k, cluto-t5-8k, cluto-t8-8k, cluto-t4-10k [11]

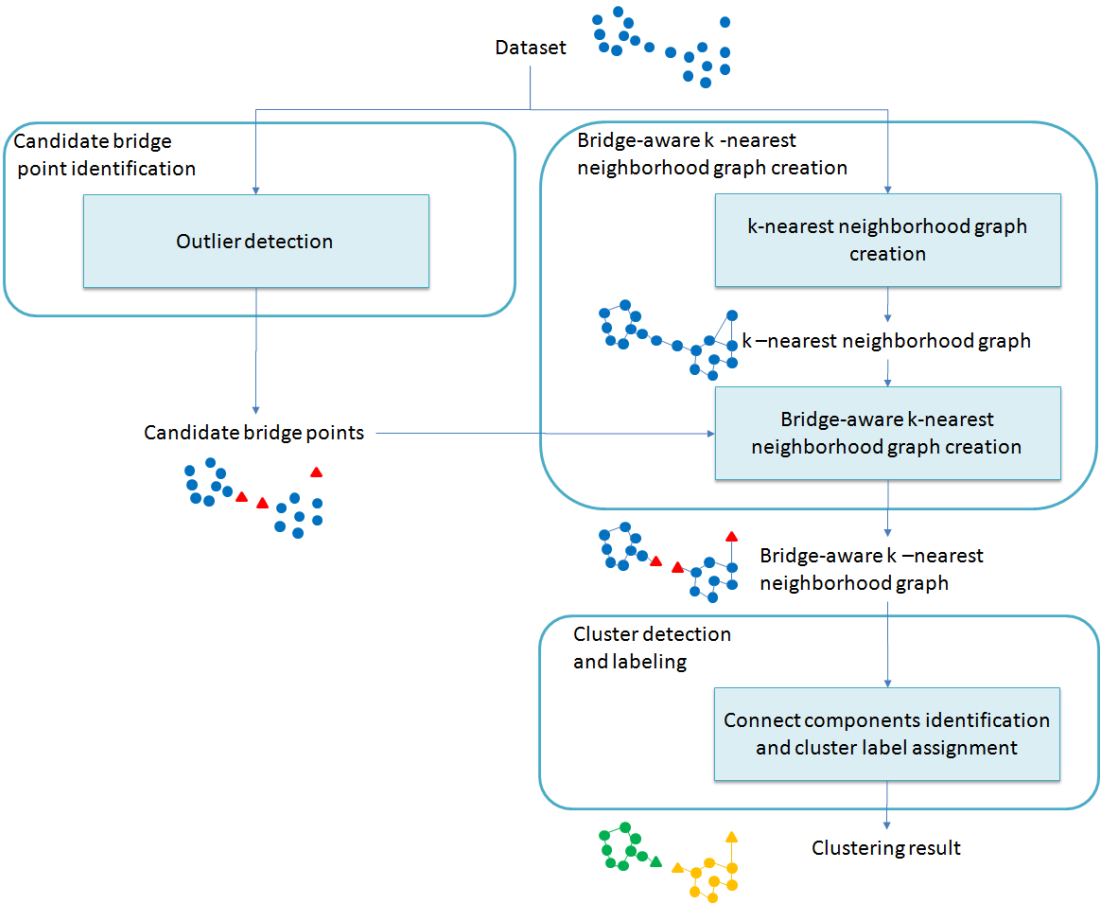


Fig. 5: The Bridge-Aware Clustering pipeline

node corresponds to a data point and outgoing edges link each point to its k-nearest neighbors. (3) In compliance with Definition 4, $\mathbb{G}$ is pruned removing the edges connecting bridge points with any other point, except for its nearest point in $\mathcal{D}$. Notice that $\mathbb{G}_{ba}$ already embeds the information about the cluster frontiers.

For example, the two clusters depicted in Figure 5 are separated by two candidate bridge points (i.e., the red triangles) detected at the previous step.

6.3 Cluster detection and labeling assignment

This step aims at producing the final clustering outcome. It defines each cluster and then labels the corresponding data point accordingly. The former step entails identifying the connected components in the bridge-aware k-nearest neighborhood graph $\mathbb{G}_{ba}$ by means of a standard connected component algorithm. Each connected component of the graph represents a separate cluster thus in the next labeling phase all nodes of the connected component are labeled with the same cluster label.

Intuitively, since the input graph is already partitioned according to the previously detected cluster frontiers, each cluster label is propagated *like a wildfire* until an outlier, acting as a candidate bridge point, is reached.

A corner case is the presence of connected components consisting of candidate bridge points only. During the cluster detection phase these connected components are not considered as separate clusters but rather as "frontiers" of other clusters. As a consequence, in the labeling phase we manage the exception by labeling these candidate bridge points as the nearest labeled point in their neighborhood.

6.4 Time complexity

We separately analyze the time complexity of each step of the BAC pipeline.

- **Outlier detection**. The complexity of LOF is $\mathrm{O}(n^2)$, where n is the number of data points [37]. Alternative outlier detection strategies have similar complexity. More details can be found here [38].
- **k-nearest neighborhood graph creation**. To discover the nearest neighbor of a data point, we exploited the scikit-learn [39] implementation of the k-NN algorithm based on the ball tree data structure [40]. To search k nearest neighbors for each of the n data points, the compexity is $\mathrm{O}(kn \cdot log(n))$.
- **Connected component finder**. Finding connected components in an undirected graph with n nodes entails visiting the graph using either breadth- or depth-first search. According to [41], the complexity is $\mathrm{O}(E)$, where E is the number of graph edges. On a fully connected graph we obtain $\mathrm{O}\left(\frac{n(n-1)}{2}\right)$, whereas for BAC the number of edges is known a priori, leading to a complexity of $\mathrm{O}(k \cdot N)$.

Overall, the time complexity of BAC is $\mathrm{O}(n^2)$. Notice that it is roughly comparable to those of many popular

density-based and density-peak algorithms. For instance, the time complexity of HDBSCAN is O(n^2), the one of DBSCAN is O(n^2) without any ad hoc data structures and decreases to O($n \cdot log(n)$) if an indexing structure is used to execute each neighborhood query in O($log(n)$). The time complexity of DADC is O($2n + C_m^2$), where C_m is the number of points after the initial pruning step.[3] Finally, Border Peeling is O($T \cdot (k \cdot n + n \cdot log(n))$, where T is the number of iterations and k is the number of considered neighbors.

7 Experimental results

This section summarizes the experimental results achieved on 45 datasets with different characteristics. Specifically, Section 7.1 describes the experimental design and the analyzed datasets, Sections 7.2 and 7.3 respectively report the outcomes of the quantitative and qualitative performance comparisons as well as the results of the statistical tests used to justify our statements, whereas Section 7.4 discusses the effect of the main algorithm parameters.

7.1 Experimental design

Environment. All the experiments were run on a Dell XPS 13 laptop with an Intel Core i7-7560U, 16GB of RAM and 512GB of SSD storage. The operating system was Ubuntu 18.04.4.

Project code and used libraries. The project code was developed and tested in Python 3.6.10. It uses the functionalities and algorithms provided by the scikit-learn 0.22.1 [39] library for data management and preparation and k-nearest neighbor definition, and by the PyOD 1.0.6 [42] library for outlier detection. The source code of this project is available for research purposes at https://github.com/lccol/bridge-clustering.

Datasets. The experiments were run on 35 synthetic benchmark datasets and 10 real-world datasets. The datasets were downloaded from the public GitHub repository [11]. A summary of the main dataset characteristics is reported in Table 1. The characteristics of the selected datasets are diversified in clusters' shape, presence of high-density and sparse regions, number of clusters, noise level, and number of features.

Competitors. We compared our approach with (i) three different traditional density-based algorithms, i.e., DBSCAN [2], HDBSCAN [4], and OPTICS [3], (ii) three density peak detection methods, i.e., DADC [9], DenMune [10], and MDPC [21], (iii) the recently proposed Border Peeling clustering algorithm [6] and AUTOCLUST [8]. For both [9] and [6] we relied on the official implementations released by the respective papers' authors. For DBSCAN and OPTICS we used the implementations available in the scikit-learn library [39]. For HDBSCAN, we used the HDBSCAN python package available at https://pypi.org/project/hdbscan/ (latest access: July 2022), whereas for AUTOCLUST we re-implemented the algorithm in Python to the best of our knowledge. Since the MDPC results are not fully reproducible, we rely on the numerical values reported on the papers and reported a separate comparison with BAC in Appendix C.

3. The order of magnitude of C_m is the same as n.

TABLE 1: Datasets summary information

Dataset	#points	#features	#clusters
Synthetic			
2d-20c-no0	1517	2	20
2d-3c-no123	715	2	3
2d-4c-no4	863	2	4
2d-4c-no9	876	2	4
aggregation	788	2	7
banana	4811	2	2
cluto-t8-8k	8000	2	9
complex8	2551	2	8
complex9	3031	2	9
cure-t0-2000n-2D	2000	2	3
cure-t1-2000n-2D	2000	2	6
cure-t2-4k	4200	2	7
dense-disk-3000	3000	2	2
diamond9	3000	2	9
disk-1000n	1000	2	2
disk-4000n	4000	2	2
disk-5000n	5000	2	2
elliptical_10_2	500	2	10
fourty	1000	2	40
golfball	4002	3	1
long1	1000	2	2
long2	1000	2	2
longsquare	900	2	6
pmf	649	3	5
smile2	1000	2	4
smile3	1000	2	4
spiralsquare	1500	2	6
triangle1	1000	2	4
triangle2	1000	2	4
twodiamonds	800	2	2
wingnut	1016	2	2
xclara	3000	2	3
zelnik1	299	2	3
zelnik5	512	2	4
zelnik6	238	2	3
Real-world			
arrhythmia	452	262	13
balance-scale	625	4	3
cpu	209	6	116
heart-statlog	270	13	2
iono	351	34	2
segment	2310	19	7
thy	215	5	3
vehicle	846	18	4
wdbc	569	31	2
zoo	101	16	7

Configuration settings. Separately for each algorithm we performed a grid search to find the configuration setting that averagely performed best on the analyzed datasets. Beyond varying the input parameter values, for non-deterministic settings we shuffled the dataset a predefined number of times to avoid introducing sampling bias.

For the Bridge-Aware Clustering algorithm we set the contamination factor of the outlier detection method to 0.2 (20% of expected noise level), the number of neighbor points to 15 and the k value for the labeling phase to 10.

The configuration settings chosen for the competitors via grid search are enumerated below.

- DBSCAN: *min points* $= 5$. Separately for each dataset we empirically set the value of ϵ using the k-distance plot and the elbow principle [16].

- HDBSCAN: *min-cluster-size* $= 15$, *min-points* $= 5$.
- OPTICS: *min-points* $= 7\%$.
- BorderPeeling: k=20.
- DADC: k=5%, $CFD\ thr$=0.6.
- DenMune: k=20.

Evaluation metric. We compared algorithm performance in terms of the Adjusted Rand Index (ARI) [36], which is a commonly used cluster evaluation measure.

7.2 Performance comparison

We quantitatively compare the performance of BAC with that of the existing algorithms. Table 2 reports the per-dataset and mean results for all the tested algorithms.[4]

BAC performed best on 25 datasets out of 45 (on 8 of them on par with other algorithms). The comparisons between the respective mean and median values confirmed the effectiveness of the proposed approach (e.g., mean ARI values: BAC 0.738 vs. HDBSCAN 0.645).[5]

Focusing the comparison on the subset of datasets characterized by either multi-density regions or noisy borders (i.e., complex9, 2d-4c-no4, 2d-20c-no0, longsquare), the performance gap is significant (mean ARI values: BAC 0.993, BP 0.7815, HDBSCAN 0.8348).

To verify the statistical significance of the performance gaps between BAC and the other algorithms, we applied the Friedman and Nemenyi statistical tests [44].[6] Specifically, we first computed the achieved rank of each algorithm per dataset. In case of ties, the average rank is reported. Then, we computed the p-value. For all the performed comparisons, the p-value is below 0.0001 thus the null hypothesis is rejected at 95% significance level. The mean rankings of each algorithm are summarized in Table 3.

We then compared the mean ranking R_i achieved by BAC with those of each of the other clustering methods. To evaluate the significance of the differences between the clustering algorithms' performance, we used the Nemenyi test. The idea behind it is to verify whether the pairwise rank differences between BAC and the considered competitor are above the critical difference (CD). According to [44], the critical difference was computed as follows.

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}}$$

Since the average ranking differences between BAC and the competitors are all greater than the critical difference ($CD = 1.124$), then we can conclude that BAC performed statistically better than all the tested competitors.

7.3 Qualitative clustering comparison

We report here some examples of clustering results achieved on datasets characterized by different data distributions and clusters' shapes.

4. The missing values for DADC and AUTOCLUST, denoted by N/A, are due to unsuccessful runs of the codes.

5. For the sake of completeness, we also reported the mean results over the 36 datasets on which all competitors' runs succeeded.

6. We omitted DADC and AUTOCLUST from the test because the algorithms did not succeed in clustering all the tested datasets. However, the difference between BAC, DADC and AUTOCLUST, in terms of average ARI, is notable (0.865 vs. 0.145 vs. 0.601).

Figure 1 shows the results achieved on the complex9 dataset by different clustering methods. As discussed in Section 2, complex9 is representative of a data space including clusters with various shapes and intra-cluster multiple densities. State-of-the-art approaches encountered issues in handling such a peculiar distribution, whereas BAC, thanks to a correct identification of a superset of the bridge points, properly separates all the clusters in the ground truth.

Figure 6 reports the results obtained on a representative dataset consisting of clusters of multiple densities (dataset 2d-4c-no4). Thanks to correct bridge identification, the proposed BAC algorithm (see Figure 6b) correctly defines the points according to the ground truth (see Figure 6a). Conversely, the other methods (see Figures 6c-6i) partly misclassified the points belonging to the sparsest clusters.

Figure 7 compares the results achieved on a dataset consisting of several small clusters, many of them characterized by noisy borders. BAC turned out to effectively cope with such a challenging data distribution (ARI = 0.994) whereas the other approaches tend to misclassify noisy points around the clusters' borders.

Figure 8 compares the results obtained on a dataset where two pairs of clusters are linked by few points in a row, which BAC correctly labeled as bridges (as the points' type evokes). BP achieved high-quality results as well, whereas all the other density-based approaches, mainly relying on core point/density peak identification (rather than on border or bridge points), failed to separate at least one of the connected cluster pairs.

7.4 Effects of parameter setting

The setup of the BAC pipeline entails configuring both the anomaly detection algorithm and the cluster labeling phase. Hence, with the twofold aim at verifying the practical usability of the proposed approach and recommending to end-users the most appropriate settings, we empirically analyzed the sensitivity of BAC performance to its input parameter values.

We reported here the results achieved by three representative unsupervised outlier detection methods: (i) Contextual: Local Outlier Factor (LOF) [37], K-Nearest neighbors (KNN) [45], [46]. (ii) Global: Histogram-based Outlier Score (HBOS) [31]. Due to the lack of space, a more extensive comparison among outlier detection algorithms is given in Appendix A. All of the aforesaid methods require to set the *contamination factor* value, which indicates the expected percentage of outliers in the analyzed dataset. The complete set of tested algorithms parameters and values is given in Table 4.

Figures 9a, 9b and 9c show the ARI value of the clustering outcomes, averaged over all the tested datasets, separately for each of the tested outlier detection algorithms by varying the k value for the cluster labeling step between 3 and 15. For the outlier detection methods we considered the following settings: $contamination = 0.2$, *n. neighbors* $= 10$ (kNN). *n. neighbors* $= 15$ (LOF). *n. bins* $= 15$ (HBOS).

Setting lower k values likely produces higher cluster fragmentation, whereas higher k values tend to underestimate the actual number of clusters. The best k values are in range [7,14] and depend on the chosen algorithm. We recommend to set k to 10, whenever not otherwise specified.

TABLE 2: Performance comparison between different clustering algorithms in terms of Adjusted Rand Score. Separately for each dataset, the ARI score of the best performing methods are written in boldface.

	BAC (our)	DBSCAN [2]	HDBSCAN [4]	BP [43]	DenMune [10]	AUTOCLUST [8]	OPTICS [3]	DADC [9]
Synthetic dataset								
2d-20c-no0	**0.994**	0.958	0.973	0.956	0.976	0.834	0.162	0.260
2d-3c-no123	0.979	0.772	0.783	0.977	**0.984**	0.795	0.787	0.158
2d-4c-no4	**0.996**	0.375	0.743	0.909	0.984	0.852	0.836	0.582
2d-4c-no9	0.966	0.785	0.909	0.802	**0.971**	0.787	0.947	0.539
aggregation	0.991	0.806	0.809	**0.993**	0.990	0.733	0.832	0.194
banana	0.876	0.955	**1.000**	0.061	0.726	0.714	0.597	N/A
cluto-t8-8k	**0.945**	0.636	0.636	0.140	0.868	0.883	0.000	0.266
complex8	0.914	**0.984**	0.372	0.339	0.703	0.816	0.216	0.178
complex9	**0.995**	0.881	0.681	0.343	0.984	0.820	0.251	0.194
cure-t0-2000n-2D	**1.000**	0.953	**1.000**	0.105	**1.000**	0.814	**1.000**	0.000
cure-t1-2000n-2D	0.958	0.978	0.885	0.234	**0.993**	0.909	0.990	0.000
cure-t2-4k	0.955	0.812	0.855	0.163	**0.981**	0.886	0.280	0.000
dense-disk-3000	**0.843**	0.801	0.825	0.045	0.011	0.682	0.000	-0.017
diamond9	0.886	0.707	0.837	0.513	**0.996**	0.495	0.040	0.368
disk-1000n	0.000	-0.001	0.033	**0.071**	0.000	-0.093	0.000	0.062
disk-4000n	**1.000**	0.900	0.933	0.045	0.000	0.415	0.000	0.032
disk-5000n	0.047	0.02	-0.003	0.021	-0.046	0.051	0.000	**0.053**
elliptical_10_2	0.669	0.667	0.669	**0.929**	0.797	0.526	0.801	0.000
fourty	**1.000**	0.992	0.998	0.000	0.998	0.728	0.000	0.000
golfball	**1.000**	**1.000**	0.000	0.000	0.000	0.000	**1.000**	0.000
long1	**1.000**	0.912	0.984	0.234	0.781	0.659	0.810	0.000
long2	**1.000**	0.962	0.986	0.187	0.867	0.718	0.809	0.000
longsquare	**0.987**	0.797	0.942	0.918	0.910	0.829	0.286	0.564
pmf	**0.964**	0.952	0.964	0.944	0.617	0.860	**0.964**	0.000
smile2	**1.000**	0.998	**1.000**	0.550	0.875	0.700	0.866	0.002
smile3	**1.000**	0.907	**1.000**	0.933	0.695	0.311	0.888	0.048
spiralsquare	**0.998**	0.972	0.986	0.960	0.570	-0.017	0.510	0.000
triangle1	**1.000**	0.952	**1.000**	0.870	0.964	0.892	0.991	0.423
triangle2	**0.974**	0.515	0.939	0.659	0.952	0.794	0.966	0.396
twodiamonds	**1.000**	0.000	0.963	0.365	**1.000**	-0.000	0.941	0.073
wingnut	**1.000**	**1.000**	0.621	0.190	**1.000**	0.590	0.000	0.232
xclara	**0.982**	0.937	0.977	0.520	0.979	0.783	0.971	0.313
zelnik1	**1.000**	0.977	**1.000**	0.867	**1.000**	0.526	0.382	0.022
zelnik5	**1.000**	0.918	**1.000**	0.925	0.913	0.729	0.976	0.262
zelnik6	0.857	**1.000**	0.722	0.813	**1.000**	0.779	0.522	0.000
Real dataset								
arrhythmia	0.000	0.063	0.000	**0.119**	0.000	N/A	0.000	0.078
balance-scale	0.000	0.000	0.017	0.000	**0.035**	-0.003	0.000	0.010
cpu	0.015	0.015	**0.015**	0.014	0.007	0.003	0.007	N/A
heart-statlog	0.000	-0.002	0.000	0.001	0.012	N/A	0.000	**0.015**
iono	0.000	**0.673**	0.423	0.344	-0.042	N/A	-0.046	N/A
segment	**0.405**	0.105	0.260	0.379	0.259	N/A	0.000	N/A
thy	0.249	0.421	-0.158	0.313	0.03	**0.568**	0.547	0.008
vehicle	0.002	0.002	0.002	**0.154**	0.120	N/A	0.000	0.006
wdbc	0.012	0.009	0.01	**0.015**	0.012	N/A	0.006	-0.001
zoo	**0.764**	0.503	0.458	0.396	0.380	N/A	0.464	N/A
Mean (36 datasets)	**0.865**	0.757	0.747	0.495	0.733	0.601	0.544	0.145
Median (36 datasets)	**0.989**	0.903	0.897	0.439	0.932	0.729	0.667	0.051
Mean (All)	**0.738**	0.657	0.645	0.429	0.619	N/A	0.458	N/A
Median (All)	**0.966**	0.806	0.825	0.343	0.867	N/A	0.464	N/A

TABLE 3: Nemenyi test. Algorithms average rankings. Critical difference CD = 1.124

	BAC	HDBSCAN	DM	DBSCAN	BP	OPTICS
Mean Rank	**2.07**	3.22	3.31	3.84	4.07	4.49

The heatmaps in Figures 9d, 9e and 9f show the ARI values, averaged over all the tested datasets, by setting a fixed k value and by varying the values of the main parameters of the outlier detection methods. LOF turned out to be the best performing method (e.g., average ARI=0.74 with *contamination*=0.2, *n. neighbors*=15 and k=10).

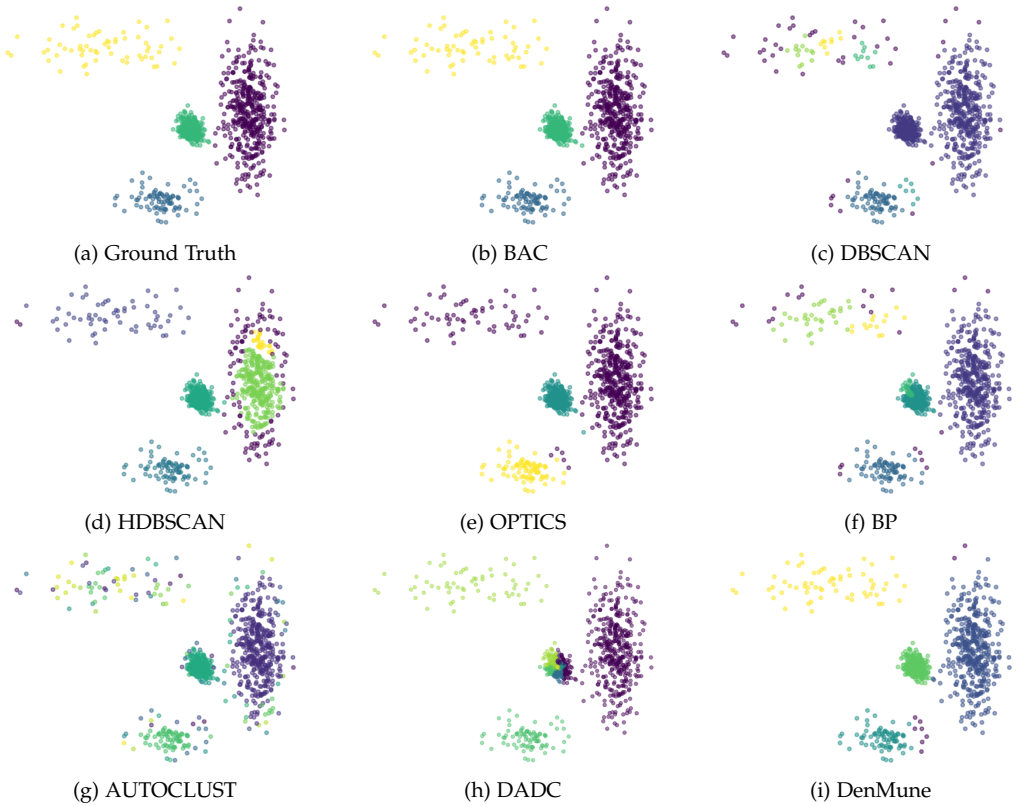
7.5 Scalability

We tested the BAC scalability by varying the number of input data points and descriptive features, respectively. The experiments were run on the synthetic blob datasets generated by means of the scikit-learn library [39].

We varied the dimensionality between 2 and 1,000 and the number of data points between 100 and 150,000.[7]

Figures 10 and 11 show the variation of the BAC's execution time. As expected, BAC scales approximately

7. Default setting: $k = 10$ (LOF), $n = 20,000$, $d = 2$. Number of independent runs: 5



(a) Ground Truth (b) BAC (c) DBSCAN

(d) HDBSCAN (e) OPTICS (f) BP

(g) AUTOCLUST (h) DADC (i) DenMune

Fig. 6: Qualitative comparison of clustering outcomes. Multi-density dataset (2d-4c-no4).

TABLE 4: Parameters' values used for the grid search.

Technique	Parameter	Values
All	contamination factor	0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.2, 0.3, 0.4, 0.5
LOF KNN	num. of neighbors	3, 5, 7, 10, 12, 15, 20
HBOS	num. of bins	5, 10, 15, 20, 25, 30, 35, 40, 45, 50
BAC	k	3, 4, 5, 7, 10, 12, 15

quadratically with the number of points and linearly with the number of input features (see Section 6.4).

8 Conclusions and Future Work

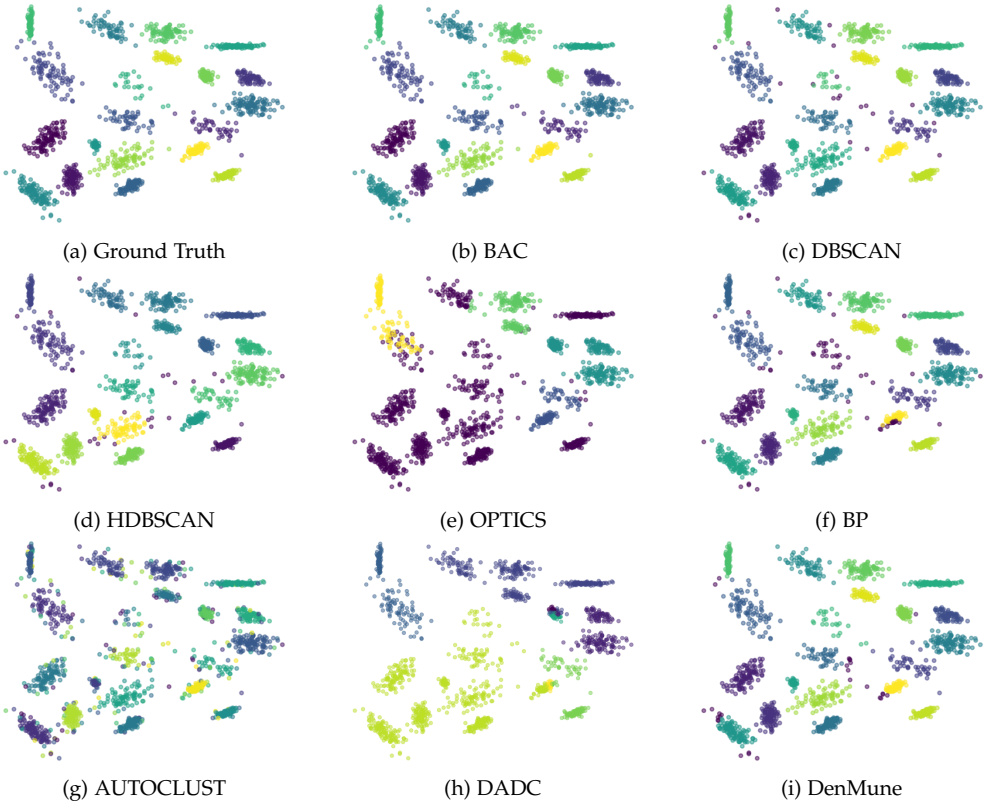
We presented a novel density-based clustering pipeline relying on the concept of bridge point. Bridge points are a newly proposed category of data points characterizing the frontier between separate clusters.

Takeaways.

- Outlier detection methods have a high recall in identifying the candidate bridge points (see Figure 4). In other words, they recognize most of the actual bridge points despite their number is often overestimated. Wrongly labeling a generic point as a bridge point potentially yields a split of some actual clusters. However, this happens only when the two sub-regions of a cluster have some points in between and all these points are erroneously labeled as candidate bridge points. If at least one of the in-between points is not labeled as a candidate bridge point, the two sub-regions of the cluster remain connected thus the outcome is not biased.
- The ARI scores of the BAC's clusters are significantly better than those achieved by state-of-the-art approaches (see Tables 2 and 3).
- Thanks to the propagation mechanism of the cluster labels BAC has shown to be robust to the presence of noisy borders and multi-density regions (see Figures 6 and 7).

Future directions.

- Firstly, we will address the problem of clustering high-dimensional data. Density-based algorithms are likely to suffer from the curse of dimensionality. To tackle the above issue, we plan to extend BAC by leveraging ad hoc embedding techniques, e.g., [47]. Furthermore, we also plan to design BAC versions suited to Big Data.
- Secondly, we plan to face the presence of noisy data by integrating ad hoc pruning strategies prior to/during the outlier detection step.
- Thirdly, we aim at adapting the proposed approach to a semi-supervised context, where both outlier detection and cluster labeling steps could benefit from partly annotated data.
- Fourthly, we plan to improve our algorithm by exploiting spectral graph theory (e.g., [47], [48]).



(a) Ground Truth (b) BAC (c) DBSCAN

(d) HDBSCAN (e) OPTICS (f) BP

(g) AUTOCLUST (h) DADC (i) DenMune

Fig. 7: Qualitative comparison of clustering outcomes. Dataset consisting of small clusters with noisy borders (2d-20c-no0).

REFERENCES

[1] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques, 3rd edition*. Morgan Kaufmann, 2011. [Online]. Available: http://hanj.cs.illinois.edu/bk3/

[2] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, vol. 96, no. 34, 1996, pp. 226–231.

[3] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: ordering points to identify the clustering structure," *ACM Sigmod record*, vol. 28, no. 2, pp. 49–60, 1999.

[4] R. J. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in *Pacific-Asia conference on knowledge discovery and data mining*. Springer, 2013, pp. 160–172.

[5] X. Li, Y. Ye, M. J. Li, and M. K. Ng, "On cluster tree for nested and multi-density data clustering," *Pattern Recognition*, vol. 43, no. 9, pp. 3130–3143, 2010.

[6] H. Averbuch-Elor, N. Bar, and D. Cohen-Or, "Border-peeling clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 7, pp. 1791–1797, 2020.

[7] J. Huang, Q. Zhu, L. Yang, D. Cheng, and Q. Wu, "Qcc: A novel clustering algorithm based on quasi-cluster centers," *Mach. Learn.*, vol. 106, no. 3, p. 337–357, Mar. 2017.

[8] V. Estivill-Castro and I. Lee, "Autoclust: Automatic clustering via boundary extraction for mining massive point-data sets," in *Proceedings of the 5th International Conference on Geocomputation*, 2000.

[9] J. Chen and P. Yu, "A domain adaptive density clustering algorithm for data with varying density distribution," *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[10] M. Abbas, A. El-Zoghabi, and A. Shoukry, "Denmune: Density peak based clustering using mutual nearest neighbors," *Pattern Recognition*, vol. 109, p. 107589, 2021.

[11] T. Barton, accessed January 5, 2021. [Online]. Available: https://github.com/deric/clustering-benchmark

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

[13] J.-G. Lee, J. Han, and K.-Y. Whang, "Trajectory clustering: A partition-and-group framework," in *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '07. ACM, 2007, p. 593–604.

[14] C. Yang, X. Shi, L. Jie, and J. Han, "I know you'll be back: Interpretable new user clustering and churn prediction on a mobile social application," *CoRR*, vol. abs/1910.01447, 2019.

[15] A. Lulli, M. Dell'Amico, P. Michiardi, and L. Ricci, "NG-DBSCAN: Scalable density-based clustering for arbitrary data," *Proc. VLDB Endow.*, vol. 10, no. 3, p. 157–168, Nov. 2016.

[16] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to data mining*. Pearson Education India, 2016.

[17] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognit. Lett.*, vol. 31, no. 8, pp. 651–666, 2010.

[18] M. Du, S. Ding, and H. Jia, "Study on density peaks clustering based on k-nearest neighbors and principal component analysis," *Knowl. Based Syst.*, vol. 99, pp. 135–145, 2016.

[19] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, 2014.

[20] M. Yan, Y. Chen, X. Hu, D. Cheng, Y. Chen, and J. Du, "Intrusion detection based on improved density peak clustering for imbalanced data on sensor-cloud systems," *Journal of Systems Architecture*, vol. 118, p. 102212, 2021.

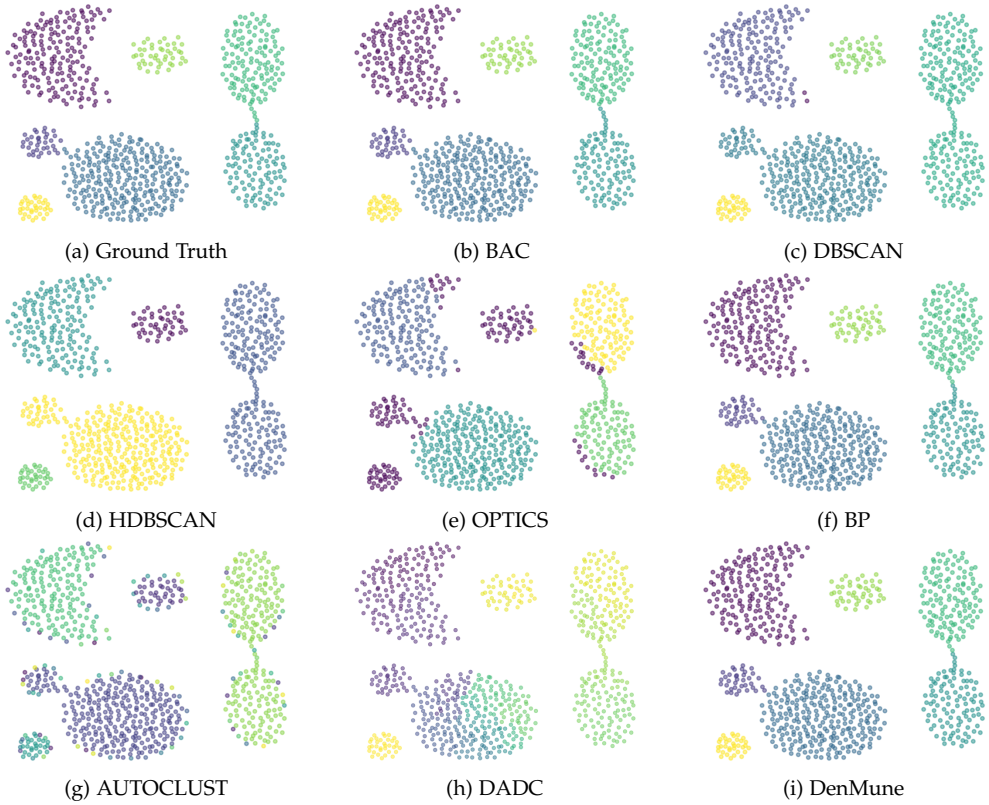
[21] X. Tao, W. Guo, C. Ren, Q. Li, Q. He, R. Liu, and J. Zou, "Density peak clustering using global and local consistency adjustable manifold distance," *Information Sciences*, vol. 577, pp. 769–804, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025521008367

[22] Y. Zheng, Q. Guo, A. K. H. Tung, and S. Wu, "Lazylsh: Approximate nearest neighbor search for multiple distance functions with a single index," in *Proceedings of the 2016 International Conference on Management of Data, SIGMOD*. ACM, 2016, pp. 2023–2037.

[23] G. Wang and Q. Song, "Automatic clustering via outward statistical testing on density metrics," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 8, pp. 1971–1985, 2016.

[24] A. Lotfi, P. Moradi, and H. Beigy, "Density peaks clustering based

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2022.3232315

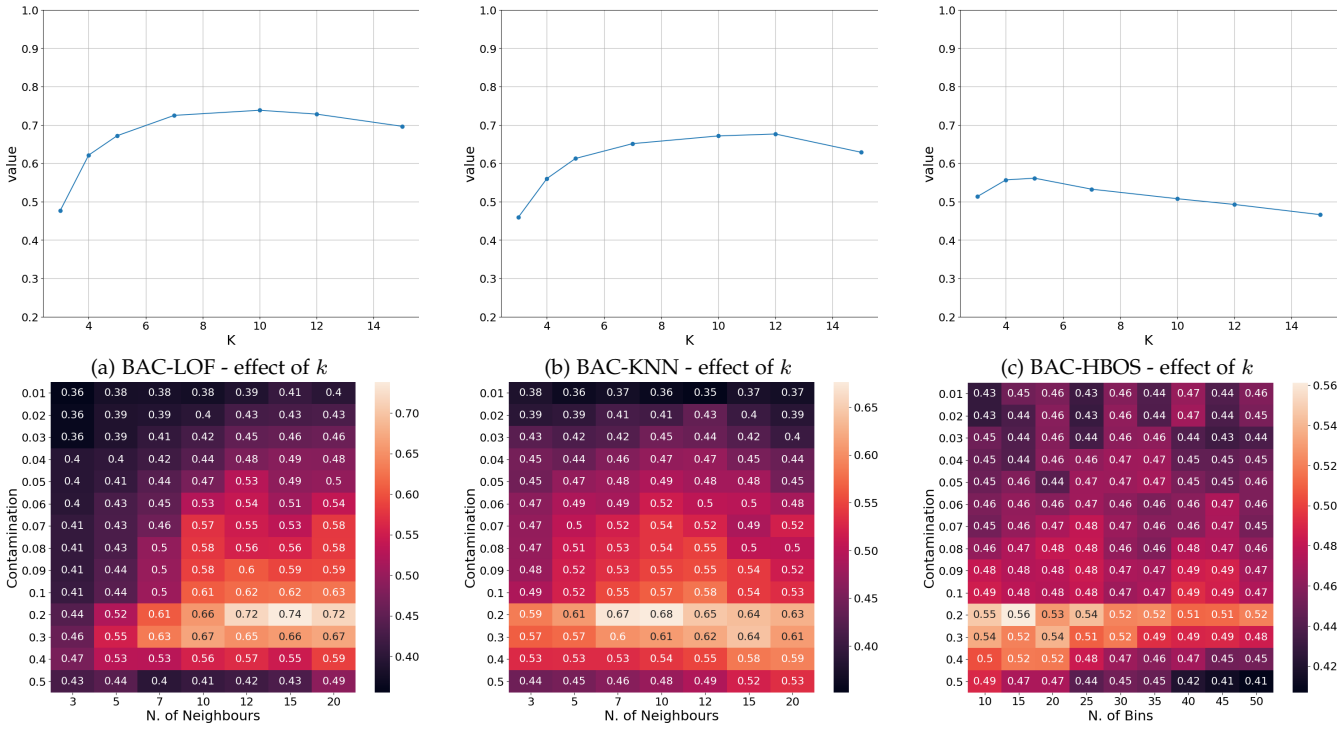


(a) Ground Truth (b) BAC (c) DBSCAN

(d) HDBSCAN (e) OPTICS (f) BP

(g) AUTOCLUST (h) DADC (i) DenMune

Fig. 8: Qualitative comparison of clustering outcomes. Clusters with connections in between (aggregation).



(a) BAC-LOF - effect of k (b) BAC-KNN - effect of k (c) BAC-HBOS - effect of k

(d) BAC-LOF - *contamination* vs. *n. neighbors* (e) BAC-KNN - *contamination* vs. *n. neighbors* (f) BAC-HBOS - *contamination* vs. *n. bins*

Fig. 9: Parameter analysis: ARI scores averaged over the 45 datasets. Outlier detection methods: LOF, KNN, HBOS.

on density backbone and fuzzy neighborhood," *Pattern Recognition*, vol. 107, p. 107449, 2020.

[25] J. Hou, A. Zhang, and N. Qi, "Density peak clustering based on relative density relationship," *Pattern Recognition*, vol. 108, p. 107554, 2020.

[26] J.-S. Lee and S. Olafsson, "Data clustering by minimizing disconnectivity," *Information Sciences*, vol. 181, no. 4, pp. 732–746, 2011.

[27] L. Liu and D. Yu, "Density peaks clustering algorithm based on

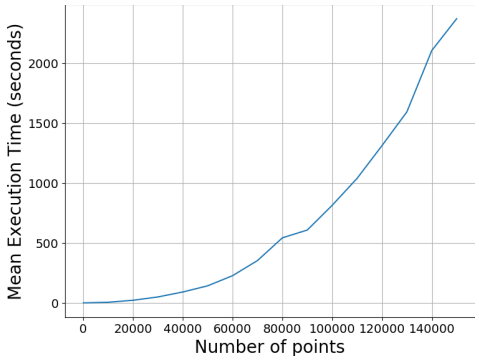


Fig. 10: BAC scalability with the number of data points.

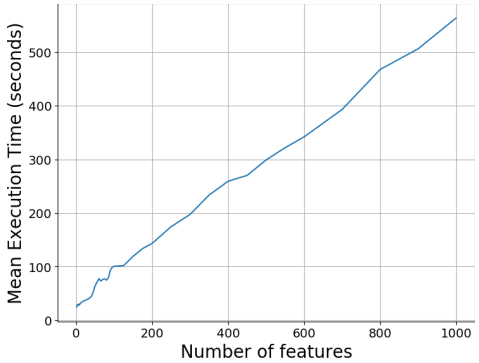
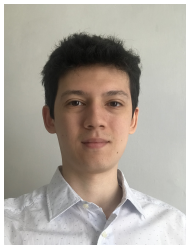


Fig. 11: BAC scalability with the number of data features.

weighted k-nearest neighbors and geodesic distance," *IEEE Access*, vol. 8, pp. 168 282–168 296, 2020.
[28] K. G. Flores and S. E. Garza, "Density peaks clustering with gap-based automatic center detection," *Knowledge-Based Systems*, vol. 206, p. 106350, 2020.
[29] Z. Yu, P. Luo, J. You, H. Wong, H. Leung, S. Wu, J. Zhang, and G. Han, "Incremental semi-supervised clustering ensemble for high dimensional data clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 3, pp. 701–714, 2016.
[30] Rui Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 645–678, 2005.
[31] M. Goldstein and A. Dengel, "Histogram-based outlier score (HBOS): A fast unsupervised anomaly detection algorithm," in *35th Annual German Conference on AI, Poster and Demo Track*, 2012, pp. 59–63.
[32] H.-P. Kriegel, M. Schubert, and A. Zimek, "Angle-based outlier detection in high-dimensional data," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008, pp. 444–452.
[33] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 2008, pp. 413–422.
[34] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support vector method for novelty detection," in *Advances in neural information processing systems*, 2000, pp. 582–588.
[35] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 665–674.
[36] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical association*, vol. 66, no. 336, pp. 846–850, 1971.
[37] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000, pp. 93–104.
[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg,
[38] O. Alghushairy, R. Alsini, T. Soule, and X. Ma, "A review of local outlier factor algorithms for outlier detection in big data streams," *Big Data and Cognitive Computing*, vol. 5, no. 1, 2021.
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, pp. 2825–2830, 2011.
[40] S. M. Omohundro, *Five balltree construction algorithms*. International Computer Science Institute Berkeley, 1989.
[41] C. H. Papadimitriou, "On the complexity of edge traversing," *J. ACM*, vol. 23, no. 3, p. 544–554, Jul. 1976.
[42] Y. Zhao, Z. Nasrullah, and Z. Li, "PyOD: A Python Toolbox for Scalable Outlier Detection," *Journal of Machine Learning Research*, vol. 20, no. 96, pp. 1–7, 2019.
[43] Chenyi Xia, W. Hsu, M. L. Lee, and B. C. Ooi, "Border: efficient computation of boundary points," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 3, pp. 289–303, 2006.
[44] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research*, vol. 7, no. Jan, pp. 1–30, 2006.
[45] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000, pp. 427–438.
[46] F. Angiulli and C. Pizzuti, "Fast outlier detection in high dimensional spaces," in *European conference on principles of data mining and knowledge discovery*. Springer, 2002, pp. 15–27.
[47] Z. Li, F. Nie, X. Chang, L. Nie, H. Zhang, and Y. Yang, "Rank-constrained spectral clustering with flexible embedding," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 12, pp. 6073–6082, 2018.
[48] Z. Li, F. Nie, X. Chang, Y. Yang, C. Zhang, and N. Sebe, "Dynamic affinity graph construction for spectral clustering using multiple features," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 12, pp. 6323–6332, 2018.



Luca Colomba received the master's degree in Computer Engineering from Politecnico di Torino in 2019, where he is currently a Ph.D. student in the Department of Control and Computer Engineering (DAUIN). His major research interests include big data analytics, scalable algorithms, data mining and machine learning applied to spatio-temporal data.

Luca Cagliero has been associate professor at the Dipartimento di Automatica e Informatica of the Politecnico di Torino since January 2020. His current research interests are in the fields of pattern mining and Deep Natural Language Processing. Specifically, he has worked on text summarization, classification and association rule mining. He has published 100+ papers in international journals, book chapters, and conference proceedings.

Paolo Garza received the master's and Ph.D. degrees in Computer Engineering from Politecnico di Torino in 2001 and 2005, respectively. Since December 2018, he has been an associate professor at the Dipartimento di Automatica e Informatica, Politecnico di Torino. His research interests include data mining algorithms, big data analytics, and data science. He has worked on classification, clustering, itemset mining, and scalable algorithms.