Hierarchical Density Estimates for Data Clustering, Visualization, and Outlier Detection

RICARDO J. G. B. CAMPELLO, Department of Computer Sciences, University of São Paulo, Brazil
DAVOUD MOULAVI, Department of Computing Science, University of Alberta, Canada
ARTHUR ZIMEK, Ludwig-Maximilians-Universität München, Germany
JÖRG SANDER, Department of Computing Science, University of Alberta, Canada

An integrated framework for density-based cluster analysis, outlier detection, and data visualization is introduced in this article. The main module consists of an algorithm to compute hierarchical estimates of the level sets of a density, following Hartigan's classic model of density-contour clusters and trees. Such an algorithm generalizes and improves existing density-based clustering techniques with respect to different aspects. It provides as a result a complete clustering hierarchy composed of all possible density-based clusters following the nonparametric model adopted, for an infinite range of density thresholds. The resulting hierarchy can be easily processed so as to provide multiple ways for data visualization and exploration. It can also be further postprocessed so that: (i) a normalized score of "outlierness" can be assigned to each data object, which unifies both the global and local perspectives of outliers into a single definition; and (ii) a "flat" (i.e., nonhierarchical) clustering solution composed of clusters extracted from local cuts through the cluster tree (possibly corresponding to different density thresholds) can be obtained, either in an unsupervised or in a semisupervised way. In the unsupervised scenario, the algorithm corresponding to this postprocessing module provides a global, optimal solution to the formal problem of maximizing the overall stability of the extracted clusters. If partially labeled objects or instance-level constraints are provided by the user, the algorithm can solve the problem by considering both constraints violations/satisfactions and cluster stability criteria. An asymptotic complexity analysis, both in terms of running time and memory space, is described. Experiments are reported that involve a variety of synthetic and real datasets, including comparisons with state-of-the-art, density-based clustering and (global and local) outlier detection methods.

Categories and Subject Descriptors: H.2.8 [**Database Mamagement**]: Database Applications—*Data mining*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Clustering*; I.5.3 [**Pattern Recognition**]: Clustering—*Algorithms*

General Terms: Data Mining, Clustering, Algorithms

Additional Key Words and Phrases: Density-based clustering, hierarchical and nonhierarchical clustering, unsupervised and semisupervised clustering, data visualization, outlier detection, global/local outliers

ACM Reference Format:
Ricardo J. G. B. Campello, Davoud Moulavi, Arthur Zimek, and Jörg Sander. 2015. Hierarchical density estimates for data clustering, visualization, and outlier detection. ACM Trans. Knowl. Discov. Data 10, 1, Article 5 (July 2015), 51 pages.
DOI: http://dx.doi.org/10.1145/2733381

This work was partially supported by NSERC (Canada), FAPESP (Brazil, grant #2013/18698-4 and grant #2010/20032-6), and CNPq (Brazil, grant #304137/2013-8 and grant #201239/2012-4).

Authors' addresses: R. J. G. B. Campello; email: campello@icmc.usp.br; D. Moulavi; email: moulavi@ualberta.ca; A. Zimek; email: zimek@dbs.ifi.lmu.de; J. Sander; email: jsander@ualberta.ca.
For part of this project, Ricardo J. G. B. Campello worked in the Department of Computing Science at the University of Alberta, Canada, on a sabbatical leave from the University of São Paulo at São Carlos, Brazil.

DOI: http://dx.doi.org/10.1145/2733381

1. INTRODUCTION

The notion of *density* plays an important role in statistics and in many data mining tasks as well. The fundamental idea behind density-based techniques for data analysis is that the dataset of interest represents a sample from an unknown probability density function (PDF), which describes the mechanism or mechanisms responsible for producing the observed data. The construction of an estimate of such a PDF from the observed data is a problem of particular relevance, for example, for analyzing and understanding the corresponding generating mechanism(s). This is the problem of *density estimation*, which can be tackled by using parametric or nonparametric approaches [Silverman 1986]. Parametric approaches assume that the data are drawn from a known parametric family of distributions (e.g., Normal), whose parameters are unknown and must be estimated. Differently, in nonparametric approaches "...less rigid assumptions are made..." and "...the data are allowed to speak for themselves in determining the estimate..." [Silverman 1986].

1.1. Multi-Level Mode Analysis for Clustering

In cluster analysis, there is also a contrast between parametric and nonparametric approaches. For example, from a statistical point of view, popular algorithms such as *k*-means and EMGM (Expectation Maximization for Gaussian Mixtures) correspond to a *parametric* approach in which an unknown PDF is assumed to be composed of a mixture of *k* Gaussian distributions, each of which is associated to one of the *k* clusters supposed to exist in the data (where *k* typically must be provided by the analyst) [Bishop 2006]. As a result, these algorithms produce a predetermined number of clusters that tend to be of convex (hyper-spherical or hyper-elliptical) shape. Notice that the limitation to convex-shaped clusters is also present in other traditional clustering algorithms, such as average linkage, Ward's, and related techniques [Jain and Dubes 1988], which do not make explicit use of parametric models. In common among these methods there is an underlying principle of "minimum variance", in the sense that all of them directly or indirectly seek to minimize a given measure of variance within clusters. The limitations of such methods, including their inability to find clusters of arbitrary shapes, have encouraged the development of alternative clustering paradigms and related algorithms that allow for more complex structures to be found in data [Xu and Wunsch II 2005, 2009]. Among those, *density-based clustering* [Tan et al. 2006; Ester 2009; Sander 2010; Kriegel et al. 2011] stands as a popular paradigm in which algorithms explicitly or implicitly incorporate elements from the theory of *nonparametric* density estimation [Hwang et al. 1994].

The first attempt at practical data clustering based on nonparametric density estimates seems to be the *One Level Mode Analysis* method and its hierarchical version (Hierarchical Mode Analysis—HMA) published by Wishart [1969] in the late 1960s. In that seminal article, Wishart listed thirteen clustering algorithms based on the "minimum variance" principle, which had already become widespread at that time, and elaborated on a number of objections to those algorithms, discussing why they may fail when applied to real world problems. He also elaborated on the limitations of the single-linkage model [Sneath 1957; Johnson 1967] and proposed a novel approach to the clustering problem, thereby establishing the grounds for what is nowadays known as *density-based clustering*. Indeed, the methods proposed by Wishart [1969] anticipated a number of conceptual and practical key ideas that have also been used by modern density-based clustering algorithms. Conceptually, for example, when referring to an estimate f of a given (possibly multivariate) PDF and a density threshold λ, Wishart observed that "...if f has two or more modes at the level of probability λ, then the covering will be partitioned into two or more disjoint connected subsets of points". This idea was formalized and extended later by Hartigan [1975], who defined the concepts of *density-contour clusters* and *density-contour tree*. These concepts play an important role as a formal probabilistic model for density-based clustering and, indeed, they have been explicitly or implicitly used as such by many algorithms belonging to this class.

For the sake of simplicity and without any loss of generality, let us consider that the observations (data objects) are described by a single continuous-valued variable (attribute), x, and that a bounded, continuous density function $f(x)$ is defined for each x as a value proportional to the number of points per unit volume at x. Then, according to Hartigan's model [Hartigan 1975], a *density-contour cluster* of $f(x)$ at a given density level λ is a subset $\mathbf{C} \subset \Re$ such that: (i) every $x \in \mathbf{C}$ satisfies $f(x) \geq \lambda$; (ii) $\mathbf{C}$ is connected; and (iii) $\mathbf{C}$ is maximal. The density-contour clusters at a given level λ are therefore the collection of maximal connected subsets of the *level set* defined as $\{x \mid f(x) \geq \lambda\}$. The *density-contour tree* is the tree of nested clusters that is conceptually conceived by varying the threshold λ. Notice that these concepts can be readily extended to more general domains other than continuous-valued densities in the real coordinates space.

The power of Hartigan's model is mainly due to the following reasons: (i) it allows the concept of *noise* to be modeled as those objects lying in nondense regions of the data space, that is, objects for which the density is below a certain threshold. This is of particular importance in cluster analysis as it breaks down the common yet usually unrealistic assumption that observations must belong to clusters and therefore they must all be clustered indistinctly; (ii) it allows clusters of varied shapes to be modeled as the connected components of the density level sets. Such components are not restricted to the domain of a single mode (peak) of the density function, they can possibly represent the union of multiple modes (depending on the density threshold); and (iii) it allows one to model the presence of nested clusters of varied densities in data, through the hierarchical relationships described by the density-contour tree.

For the reasons mentioned earlier, most density-based clustering algorithms are, in essence, strictly or loosely based on Hartigan's model.[1] The differences basically rely on the way the density f and the components of a level set are estimated. For Wishart's one level mode analysis method (with parameters r and κ), for example, it can be shown that the results correspond to the use of an estimator of connected components given by the union of balls of radius $r/2$, a K nearest neighbor (K-NN) density estimate f with $K = \kappa$, and a density threshold λ as a function of r (or, equivalently, f can also be seen as a kernel-based density equipped with square-wave kernels of width r and λ as a function of κ). The same holds true for the popular algorithm DBSCAN [Ester et al. 1996] if one denotes its parameters, neighborhood radius and minimum number of objects within the neighborhood, as r and κ, respectively. DBSCAN is, however, computationally more sophisticated and scalable as it makes use of efficient indexing structures. There is also a conceptual difference since, in DBSCAN, those non-dense objects that lie in the neighborhood of dense objects, the so-called *border objects*, are assigned to clusters even though their density is below the established threshold. Notice that such a strategy makes DBSCAN only loosely in conformity with Hartigan's model.[2] This is also the case for the well-known algorithm DENCLUE [Hinneburg and Keim 1998, 2003; Hinneburg and Gabriel 2007], which uses a general kernel-based density estimate as well as a gradient-based approach to decide which objects should

[1] In contrast, works such as [Fukunaga and Hostetler 1975; Coomans and Massart 1981] make the oversimplistic assumption that each mode of the density f corresponds to a cluster and, then, they simply seek to assign every data object to one of these modes (according to some hill-climbing-like heuristic).

[2] Wishart's one level mode analysis method also does not follow strictly Hartigan's model if the optional postprocessing stage of that method is performed, in which the non-dense objects are arbitrarily assigned to the "nearest" cluster.

be clustered and which ones should not (and thus be left as noise). The decision strategy is based on a density threshold, but it still allows objects whose density is below the threshold to be clustered (if they are attracted by high density modes of the density estimate).

One level mode analysis, DBSCAN, and DENCLUE are examples of algorithms that adopt Hartigan's model limited to a single threshold λ only, thus producing a "flat" clustering solution based on a global discriminating density level (rather than a tree of density-contour clusters at different levels). This approach has some fundamental limitations [Tan et al. 2006; Kriegel et al. 2011]: (i) the choice of the density threshold is critical; (ii) often, it is not possible to simultaneously detect clusters of varied densities by using a single, global density threshold; and (iii) a flat clustering solution alone cannot describe possible hierarchical relationships that may exist between nested clusters lying on different density levels. Nested clusters at varied levels of density can only be described by hierarchical density-based clustering methods, such as those in Wishart [1969], Wong and Lane [1983], Ankerst et al. [1999], Sander et al. [2003], Brecheisen et al. [2004], Chehreghani et al. [2008], Stuetzle and Nugent [2010], Sun et al. [2010], Gupta et al. [2010], and Campello et al. [2013a], which are able to provide more elaborated descriptions of a dataset at different degrees of granularity and resolution.

Hierarchical models are indeed able to provide richer descriptions of clustering structures than those provided by flat models. In spite of this, applications in which the user also needs a flat solution are common, either for further manual analysis by a domain expert or in automated KDD processes in which the output of a clustering algorithm is the input of a subsequent data mining procedure—for example, pattern recognition based on image segmentation. In this context, the extraction of a flat clustering from a hierarchy may be advantageous when compared to the extraction directly from data by a partitioning-like (i.e., nonhierarchical) algorithm. The reason is that hierarchical models describe data from multiple levels of specificity/generality, providing a means for exploration of multiple possible solutions from different perspectives while having a global picture of the cluster structure available.

1.2. Global and Local Outlier Detection

Similar as for clustering, parametric, statistical approaches for unsupervised outlier detection (identification, rejection) fit certain distributions to the data by estimating the parameters of these distributions from the given data [Grubbs 1950; Barnett 1978; Beckman and Cook 1983; Hodge and Austin 2004; Agyemang et al. 2006; Hadi et al. 2009]. For example, when assuming a Gaussian distribution, a commonly used rule of thumb, known as the "$3 \cdot \sigma$-rule", is that points deviating more than three times the standard deviation from the mean of a normal distribution may be considered outliers [Knorr and Ng 1997b]. Classical textbooks [Hawkins 1980; Barnett and Lewis 1994] discuss numerous tests for different distributions. The tests are optimized for each distribution dependent on the specific parameters of the corresponding distribution, the number of expected outliers, and the space where to expect an outlier.

A problem with parametric approaches is that distribution parameters such as mean, standard deviation, and covariances are rather sensitive to the presence of outliers. Possible effects of outliers on the parameter estimation have been termed *"masking"* and *"swamping"*. Outliers can *mask* their own presence by influencing the values of the distribution parameters (resulting in false negatives), or *swamp* inliers to appear as outlying due to the influenced parameters (resulting in false positives) [Pearson and Chandra Sekar 1936; Beckman and Cook 1983; Barnett and Lewis 1994; Hadi et al. 2009]. Furthermore, these approaches are deemed to assume a specific type of distribution.

Nonparametric approaches for outlier detection do not assume a specific distribution of the data, but estimate (explicitly or implicitly) certain aspects of the probability density. Nonparametric methods include, for instance, the well-known "distance-based" methods [Knorr and Ng 1997b; Ramaswamy et al. 2000; Angiulli and Pizzuti 2002] and also "density-based" methods such as LOF (local outlier factor) [Breunig et al. 2000] and its many variants. An important categorization of these approaches distinguishes "local" versus "global" approaches. Although this categorization is not strictly dichotomous but there are degrees of locality [Schubert et al. 2014b], these categories reflect the nature of outliers that can be found by local or by global methods. Global outliers are outliers that are unusual w.r.t. the complete database. Local outliers are outliers that are unusual w.r.t. a local selection of the database only, such as an ε-range query or the K-NNs, but that would not necessarily look suspicious when compared globally to the complete database. Not only are there degrees of locality in the design of methods, but also a local method such as LOF could be used (or rather abused) in a global way by considering the complete database as the NNs (i.e., $K =$ database size). This would be not in the spirit of the method but even so, one method with one parametrization is either local (with a certain degree) or global, but not both at the same time. It would be rather the spirit of LOF to choose the size of the neighborhood in a way to at least include some objects of the nearest cluster for an outlier while not including objects of another cluster for inliers. Choosing the right K for the neighborhood, of course, is a task unsolvable without knowing the data already. Thus, when choosing a method and a parametrization (such as neighborhood size), the users have to decide which kind of outliers, local or global, they are interested in.

The problem of having to select a particular threshold that inherently limits the types of outliers that a local method can detect in a given dataset (which may contain both global and local outliers) has also been recognized by Papadimitriou et al. [2003], who proposed the method LOCI that tries to address this problem. LOCI is probably the first, and so far only, attempt at finding for each point a neighborhood that is specific to the point in determining whether the point is an outlier or not. Depending on the distance that defines the neighborhood, a point can be a local outlier in some sense or even a global outlier. However, LOCI considers circular neighborhoods (just like LOF and related methods, yet at different distances around a point), which may not take the relevant structure of the data properly into account. Circular neighborhoods, at large scales, will typically contain "neighbors" from multiple, distant clusters whose properties can be considered as unrelated to the point at the center of the sphere, yet they are taken into account in the calculation of its outlier score.

In this article, we argue that by basing an outlier model on *hierarchical* density estimates, one can adapt not only to locally different density-levels (such as LOF and its variants) but also to locally different notions of "local". Traditionally, each query object's outlierness is assessed w.r.t. some reference set (such as a *local*, circular neighborhood or the *global* database). In our approach, for each query object, the scope of a reference set of objects is chosen dynamically and based on the *closest structure* within the density-based hierarchy. By doing so, the locally suitable choice could actually be global for some objects while being strictly local for other objects. In any case, the most meaningful reference structure for a point is the "nearest" cluster in the hierarchical cluster hierarchy, as opposed to all points in some sphere around the point. This avoids comparing outliers to other outliers or unrelated clusters, as it could happen for a local method or LOCI; it also enables the user to find both global and local outliers and be able to relate them to the overall clustering structure within one efficient approach. Compared to previous methods, our approach represents somewhat of a paradigm shift in what constitutes meaningful neighborhoods for the determination of outlier scores. However, we argue that the outlier model we propose is in fact more in line with the widely cited definition of outliers by Hawkins (an outlier is *"an observation that deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism"* [Hawkins 1980]) and is based on solid statistical modeling without suffering from the typical problems of parametric approaches.

Some authors criticize the use of clustering algorithms as an approach to detect outliers [Knorr and Ng 1998; Ramaswamy et al. 2000; Knorr et al. 2000; Tang et al. 2002; Kriegel et al. 2010; Zimek et al. 2012]. It is worth noting, though, that such criticisms refer to algorithms that produce "outliers" as ordinary by-products of the clustering procedure. Typical examples are those data objects left unclustered as noise by algorithms such as DBSCAN and DENCLUE (in the latter, such objects are explicitly called "outliers" indeed). The labeling of these objects as noise, however, is a mere consequence of the density threshold arbitrarily set by the analyst (the data are not speaking for themselves). In contrast, the interrelated notions of outliers and clusters we introduce in this article depend on the hierarchy of density estimates and its shape as a whole, rather than on a single "snapshot" taken at a particular, arbitrary level. We advocate that the problems of clustering and outlier detection should be treated simultaneously rather than separately (provided that this can be done properly) for two reasons: (i) when referring to an outlier as *an observation that deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism*, as in the classical definition by Hawkins [1980], one is obviously assuming the existence of one or more mechanisms responsible for generating the observations deemed nonsuspicious. The modeling of such mechanisms seems to be, therefore, an important (if not fundamental) step towards the detection of suspicious observations as outliers. Intuitively, clusters are natural candidates for this task; and (ii) an outlier is not always viewed as an observation that deviates too much from *all* the other observations. In fact, in some application domains outliers may actually refer to observations that are similar to each other for they are generated by the same mechanism, although they are typically less frequent than and deviate from other observations (e.g., certain types of frauds or genetic mutations following a common pattern). These outliers can still be modeled and detected as clusters which are typically smaller in size or volume and possibly farther apart from other clusters.

1.3. Contributions

In this article, we introduce a complete framework for density-based clustering, outlier detection, and visualization. The core of the framework is a method based on nonparametric density estimates that gives rise to a hierarchical clustering algorithm HDBSCAN* (*Hierarchical DBSCAN**). HDBSCAN* follows Hartigan's model of density-contour clusters/trees and improves existing density-based clustering algorithms w.r.t. different aspects (to be further elaborated in the related work section). It provides as a result a complete clustering hierarchy composed of all possible density-based clusters following the nonparametric model adopted, for an infinite range of density thresholds, and from which a simplified cluster tree can be easily extracted by using Hartigan's concept of *rigid clusters* [Hartigan 1975].

The proposed framework is illustrated in Figure 1. The hierarchy and the cluster tree produced by the central module (HDBSCAN*) can be postprocessed for multiple tasks. For instance, they are particularly suitable for interactive data exploration, as they can be easily transformed and visualized in different ways, such as an OPTICS reachability plot [Ankerst et al. 1999; Sander et al. 2003], a silhouette-like plot [Gupta et al. 2010], a detailed dendrogram, or a compacted cluster tree. In addition, for applications that expect a nonhierarchical partition of the data, the clustering hierarchy can also be postprocessed so that a flat solution—as the best possible nonhierarchical

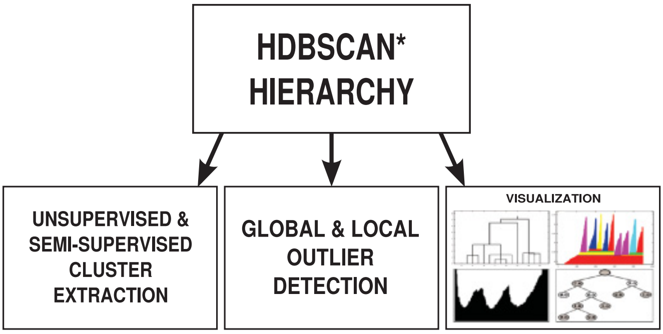

HDBSCAN* HIERARCHY
UNSUPERVISED & SEMI-SUPERVISED CLUSTER EXTRACTION
GLOBAL & LOCAL OUTLIER DETECTION
VISUALIZATION

Fig. 1. Proposed framework for hierarchical & nonhierarchical density-based clustering, global/local outlier detection, and data visualization.

representation of the data in some sense—can be extracted. The traditional approach to get a flat solution from a hierarchical clustering is to perform a global horizontal cut through one of the levels of the hierarchy, but this approach inherits all the limitations we discussed earlier regarding the use of a single density threshold. For this reason, we instead advocate the use of a method by which a flat solution composed of clusters extracted from local cuts through the cluster tree (possibly corresponding to different density thresholds) can be obtained, either in an unsupervised or even in a semisupervised way. For the unsupervised scenario, we describe an algorithm that provides a globally optimal solution to the formal problem of maximizing the overall stability of the extracted clusters, for which a cluster stability measure is formulated. If partially labeled objects or instance-level constraints of the type *should-link* and *should-not-link* are provided by the user (semisupervised scenario), the algorithm can solve the problem by considering both constraint violations/satisfactions and cluster stability criteria. Finally, besides the clustering and visualization tasks, the density-based hierarchy produced by HDBSCAN* can also be used as the basis for a novel, effective, and efficient outlier detection method, as illustrated in Figure 1. The method we propose for the outlier detection module unifies both the global and local flavors of the outlier detection problem into a single definition of an outlier detection measure, called GLOSH (*Global-Local Outlier Scores from Hierarchies*), which also attempts at reconciling those more statistically inclined and those more database oriented methods for unsupervised outlier detection, by means of a nonparametric approach.

In detail, we make the following contributions within our proposed framework:

(1) For the core module of the framework, we present and discuss in details HDBSCAN* as a hierarchical clustering method that generates a complete density-based clustering hierarchy from which a simplified cluster tree composed only of the most significant clusters can be easily extracted.
(2) We describe and discuss different ways of visualizing the HDBSCAN* results.
(3) We present a measure of cluster stability for the purpose of extracting a flat clustering solution from local cuts (possibly corresponding to different density levels) through the HDBSCAN* hierarchy; we formulate the task of extracting such a flat solution as an optimization problem in which the overall stability of the composing clusters (unsupervised scenario) and/or the fraction of instance-level constraints that are satisfied (semisupervised scenario) are maximized; and we describe an algorithm that finds the globally optimal solution to this problem.
(4) We propose GLOSH as a new, effective, and efficient outlier detection measure, which is possibly unique in that it can simultaneously detect both global and local

types of outliers, based on a sound statistical interpretation; and we describe an algorithm to efficiently compute GLOSH scores from the HDBSCAN* hierarchy.

(5) We demonstrate the advancement that our framework represents in the areas of density-based clustering and unsupervised outlier detection through extensive experiments on a variety of synthetic and real world datasets.

Note that in a preliminary publication [Campello et al. 2013a], we have very briefly introduced the HDBSCAN* algorithm and the postprocessing procedure to optimally extract a flat clustering solution from the corresponding hierarchy, in the unsupervised scenario. The current article significantly extends this work by providing detailed explanations, examples, and complexity analyses. In another previous publication [Campello et al. 2013b], we introduced a general framework to optimally extract flat clustering solutions from clustering hierarchies, in a semisupervised way that encompasses the unsupervised scenario as a special case. This framework, however, has to be specialized and instantiated for use with each different type of hierarchy. The current article introduces and experimentally evaluates a new specialization for the HDBSCAN* hierarchy. Finally, the work described here on outlier detection presents completely new ideas and original material that has not been introduced elsewhere.

1.4. Outline of the Article

The remainder of this article is organized as follows. In Section 2, we redefine DBSCAN in a way that makes it more consistent with Hartigan's model. Then, in Section 3, we present and provide an extensive description and discussion of the algorithm HDBSCAN*. In Section 4, we discuss some different alternatives for the visualization of the HDBSCAN* results. In Section 5, we pose the problem of extracting a nonoverlapping collection of clusters from the HDBSCAN* hierarchy as an optimization problem, and describe an algorithm to solve this problem in an unsupervised or semisupervised way. In Section 6, we propose GLOSH, a novel outlier detection method, and describe a simple procedure to compute the corresponding outlier scores from the HDBSCAN* hierarchy. We discuss related work in Section 7. In Section 8, we present an extensive experimental evaluation involving real and synthetic data as well as comparisons with state-of-the-art algorithms for density-based clustering and for global and local outlier detection. Section 9 concludes the article.

2. DBSCAN REVISITED—THE ALGORITHM DBSCAN*

Let $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be a dataset containing n data objects, each of which is described by an attribute vector, $\mathbf{x}_{(\cdot)}$. In addition, let D be (*conceptually* only) an $n \times n$ symmetric matrix containing the distances $d(\mathbf{x}_p, \mathbf{x}_q)$ between pairs of objects of $\mathbf{X}$ in a metric space[3]. In the following, we define the density-based clustering algorithm DBSCAN* as in our preliminary work [Campello et al. 2013a], which differs from DBSCAN [Ester et al. 1996] in that the clusters are defined based on *core objects* alone.

Definition 2.1 (*Core and Noise Objects*). An object $\mathbf{x}_p$ is called a *core object* w.r.t. ε and m_{pts} if its ε-neighborhood contains at least m_{pts} many objects, that is, if $|\mathbf{N}_\varepsilon(\mathbf{x}_p)| \geq m_{\text{pts}}$, where $\mathbf{N}_\varepsilon(\mathbf{x}_p) = \{\mathbf{x} \in \mathbf{X} | d(\mathbf{x}, \mathbf{x}_p) \leq \varepsilon\}$ and $|\cdot|$ denotes cardinality. An object is called *noise* if it is not a core object.

Definition 2.2 (*ε-Reachable*). Two *core* objects $\mathbf{x}_p$ and $\mathbf{x}_q$ are *ε-reachable* w.r.t. ε and m_{pts} if $\mathbf{x}_p \in \mathbf{N}_\varepsilon(\mathbf{x}_q)$ and $\mathbf{x}_q \in \mathbf{N}_\varepsilon(\mathbf{x}_p)$.

[3] Matrix D is not required if distances $d(\cdot, \cdot)$ can be computed from $\mathbf{X}$ on demand.

Definition 2.3 (*Density-Connected*). Two *core* objects $\mathbf{x}_p$ and $\mathbf{x}_q$ are *density-connected* w.r.t. ε and m_{pts} if they are directly or transitively ε-reachable.

Definition 2.4 (*Cluster*). A *cluster* $\mathbf{C}$ w.r.t. ε and m_{pts} is a non-empty maximal subset of $\mathbf{X}$ such that every pair of objects in $\mathbf{C}$ is density-connected.

Based on these definitions, we can devise an algorithm DBSCAN* (similar to DBSCAN) that conceptually finds clusters as the connected components of a graph in which the objects of $\mathbf{X}$ are vertices and every pair of vertices is adjacent if and only if the corresponding objects are ε-reachable w.r.t. user-defined parameters ε and m_{pts}. Noncore objects are labeled as noise.

Note that the original definitions of DBSCAN also include the concept of *border* objects, that is, noncore objects that are within the ε-neighborhood of one or more core objects. Border objects are in DBSCAN assigned to a cluster corresponding to one of these core objects. When using DBSCAN*, one could also include the border objects in a simple, linear time postprocessing step (tracking and assigning each border object to, e.g., its closest core). However, our new definitions are more consistent with a statistical interpretation of clusters as connected components of a level set of a density [Hartigan 1975], since border objects do not technically belong to the level set (their estimated density is below the threshold). The new definitions also imply that clusters are formed based on a symmetric notion of reachability that allows a precise relationship between DBSCAN* and its hierarchical version, to be discussed in the next section. This was only approximately possible between DBSCAN and OPTICS [Ankerst et al. 1999][4]; and including border objects also complicated the formulation of the OPTICS algorithm, preventing at the same time a precise statistical interpretation.

3. HIERARCHICAL DBSCAN*—HDBSCAN*

In this section, we provide an extended description and discussion of our hierarchical clustering method, HDBSCAN* [Campello et al. 2013a], which can be seen as a conceptual and algorithmic improvement over OPTICS [Ankerst et al. 1999].

3.1. Conceptual HDBSCAN*

Our method has as its single input parameter a value for m_{pts}. This is a classic smoothing factor in density estimates whose behavior is well understood, and methods that have an analogous parameter (e.g., Ankerst et al. [1999], Gupta et al. [2010], Pei et al. [2009], and Stuetzle and Nugent [2010]) are typically robust to it.

For a proper formulation of the density-based hierarchy w.r.t. a value of m_{pts}, we employ notions related to the *core* and *reachability distances* introduced for OPTICS. While the notion of core distance is the same as for OPTICS, we use, however, a symmetric definition of reachability distance ("*mutual* reachability distance"), following the definition of Lelis and Sander [2009].

Definition 3.1 (*Core Distance*). The *core distance* of an object $\mathbf{x}_p \in \mathbf{X}$ w.r.t. m_{pts}, $d_{\text{core}}(\mathbf{x}_p)$, is the distance from $\mathbf{x}_p$ to its m_{pts}-nearest neighbor (including $\mathbf{x}_p$).

Notice that the core distance is the minimum radius ε such that $\mathbf{x}_p$ satisfies the core condition w.r.t. m_{pts}, that is, $|\mathbf{N}_\varepsilon(\mathbf{x}_p)| \geq m_{\text{pts}}$ (Definition 2.1).

[4] Ankerst et al. [1999] notice that flat clusterings extracted from an OPTICS reachability plot are "*nearly* indistinguishable from a clustering created by DBSCAN".

Definition 3.2 (*Mutual Reachability Distance*). The *mutual reachability distance* between two objects $\mathbf{x}_p$ and $\mathbf{x}_q$ in $\mathbf{X}$ w.r.t. m_{pts} is defined as $d_{\text{mreach}}(\mathbf{x}_p, \mathbf{x}_q) = \max\{d_{\text{core}}(\mathbf{x}_p), d_{\text{core}}(\mathbf{x}_q), d(\mathbf{x}_p, \mathbf{x}_q)\}$.

Notice that $d_{\text{mreach}}(\mathbf{x}_p, \mathbf{x}_q)$ in Definition 3.2 is the minimum radius ε such that $\mathbf{x}_p$ and $\mathbf{x}_q$ are ε-reachable according to Definition 2.2. It plays a central role in defining the following (*conceptual* only) transformed proximity graph, which will help us explain a density-based clustering hierarchy.

Definition 3.3 (*Mutual Reachability Graph*). The *mutual reachability graph* is a complete graph, $G_{m_{\text{pts}}}$, in which the objects of $\mathbf{X}$ are vertices and the weight of each edge is the mutual reachability distance (w.r.t. m_{pts}) between the respective pair of objects.

Let $G_{m_{\text{pts}},\varepsilon} \subseteq G_{m_{\text{pts}}}$ be the graph obtained by removing all edges from $G_{m_{\text{pts}}}$ having weights greater than some value of ε. From Definitions 2.4 and 3.3, it is straightforward to infer that clusters according to DBSCAN* w.r.t. m_{pts} and ε are then the connected components of core objects in $G_{m_{\text{pts}},\varepsilon}$; and the remaining objects are noise. Consequently, all DBSCAN* clusterings for any $\varepsilon \in [0, \infty)$ can be produced in a nested, *hierarchical* way by removing edges in decreasing order of weight from $G_{m_{\text{pts}}}$.

At this point, it is important to notice that removing edges with weights greater than a decreasing threshold from a complete proximity graph, and then checking for the remaining connected subcomponents of the graph, is essentially the graph-based definition of the hierarchical Single-Linkage algorithm [Johnson 1967] (e.g., refer to Jain and Dubes [1988] for details). The following proposition then holds, which formalizes the conceptual relationship between the algorithms DBSCAN* and Single-Linkage in the transformed space of mutual reachability distances.

Proposition 3.4. *Let $\mathbf{X}$ be a set of n objects described in a metric space by $n \times n$ pairwise distances. The clustering of this data obtained by DBSCAN* w.r.t m_{pts} and some value ε is identical to the one obtained by first running Single-Linkage over the transformed space of mutual reachability distances (w.r.t m_{pts}), then, cutting the resulting dendrogram at level ε of its scale, and treating all resulting singletons with $d_{\text{core}}(\mathbf{x}_p) > \varepsilon$ as a single class representing "Noise".*

Proof. Proof sketch as per discussion earlier, after Definition 3.3. □

Corollary 3.5. *For $m_{\text{pts}} = 1$ or $m_{\text{pts}} = 2$, DBSCAN* w.r.t. ε, m_{pts} is equivalent to a horizontal cut through level ε of the Single-Linkage dendrogram in the original space of distances $d(\cdot, \cdot)$, provided that all resulting singletons are labeled as noise if $m_{\text{pts}} = 2$ or as unitary clusters if $m_{\text{pts}} = 1$.*

Proof. From Definition 3.1, the core distance of an object $\mathbf{x}_p$ is equal to zero when $m_{\text{pts}} = 1$ and is equal to the distance to its NN (excluding the object itself) when $m_{\text{pts}} = 2$. Accordingly: (i) for $m_{\text{pts}} = 1$, any "isolated" (i.e., nonconnected) object at any level ε is necessarily a core object and, therefore, a unitary cluster at that level. In other words, no noise will exist in this case, no matter the value of ε; (ii) for $m_{\text{pts}} = 2$, it follows from Definitions 3.1 and 3.2 that $d_{\text{core}}(\mathbf{x}_p)$ is equal to the minimum $d_{\text{mreach}}(\mathbf{x}_p, \mathbf{x}_q)$ between $\mathbf{x}_p$ and any other object $\mathbf{x}_q$. This means that, for a given cut level ε, any isolated object will necessarily be a noise object at that level; and (iii) for both $m_{\text{pts}} = 1$ and $m_{\text{pts}} = 2$, it follows from Definition 3.2 that $d_{\text{mreach}}(\mathbf{x}_p, \mathbf{x}_q) = d(\mathbf{x}_p, \mathbf{x}_q)$, that is, the original and transformed data spaces are the same for these particular values of m_{pts}. As a consequence, the equivalence between DBSCAN* and Single-Linkage described in Proposition 3.4 is valid even for the original space when $m_{\text{pts}} = 1$ or $m_{\text{pts}} = 2$ (the use of larger values of m_{pts}, on the other hand, is known to weaken the chaining effect of Single-Linkage by stretching the distances between objects lying in sparser regions of the data space). □

Proposition 3.4 suggests that we could implement a hierarchical version of DBSCAN* by applying an algorithm that computes a Single-Linkage hierarchy on the *transformed* space of mutual reachability distances. The simple application of Single-Linkage to the transformed distance space, however, would not directly encode whether an isolated object is a core or a noise object at a given level of the hierarchy. As previously discussed, a density-based cluster hierarchy has to represent the fact that an object o is noise below the level l that corresponds to o's core distance. Instead of a postprocessing of the resulting hierarchy to include this information, we adopt a more efficient and more elegant solution.

3.2. Algorithm HDBSCAN*

The fastest way to compute a Single-Linkage hierarchy is possibly by using a divisive algorithm based on the Minimum Spanning Tree (MST) [Jain and Dubes 1988], which works by removing edges from an MST in decreasing order of weights (here, corresponding to mutual reachability distances). To directly represent the level in the hierarchy below which an isolated object o is a noise object, we need to include an additional node at that level, representing the cluster containing the single object o at that level and higher. This can be achieved by extending the MST with self-loops (edges connecting each vertex to itself), in which the edge weight for each vertex o is set to the core distance of o; these "self-edges" will then be considered when removing edges. Notice from Definition 3.2 that the weight of a self-loop cannot be greater than those of the other edges incident to the corresponding vertex. In case of ties, edges are removed simultaneously.

Algorithm 1 shows the pseudocode for HDBSCAN*. It has as inputs the value for m_{pts} and either $\mathbf{X}$ or the corresponding distance matrix D. For a given m_{pts}, the algorithm produces a clustering tree that contains all clusterings obtainable by DBSCAN* in a hierarchical, nested way, including nodes that indicate when an isolated object changes from core (i.e., dense) to noise. The result is called the "HDBSCAN* hierarchy". The set of all unique clusterings that can be obtained by DBSCAN*, for a value of m_{pts} and all possible values of the radius ε, corresponds one-to-one to the set of unique scale values (MST_{ext} edge weights) of the hierarchy levels. When combining consecutive levels of the hierarchy with the same scale value (ties of the MST_{ext} edge weights), as usual in classic Single-Linkage dendrograms, the levels themselves correspond one-to-one to the unique clusterings obtainable by DBSCAN*.

3.3. Hierarchy Simplification

A dendrogram is an important exploratory data analysis tool, but in its raw form it may be difficult to interpret or process it for large and "noisy" datasets. In this context, it is an important task to extract from a dendrogram a summarized tree of only "significant" clusters. We propose a simplification of the HDBSCAN* hierarchy based on a fundamental observation about estimates of the level sets of continuous-valued PDF, which refers back to Hartigan's concept of *rigid clusters* Hartigan [1975], and which has also been employed similarly by Gupta et al. [2010]. For a given PDF, there are only three possibilities for the evolution of the connected components of a continuous density level set when increasing the density level (decreasing ε in our context) [Herbin et al. 2001]: (i) the component shrinks but remains connected, up to a density threshold at which either (ii) the component is divided into smaller ones, or (iii) it disappears.

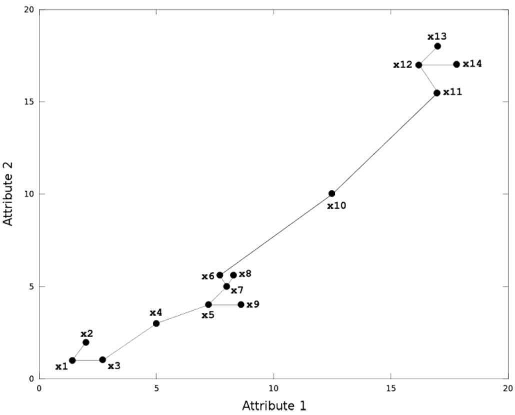


Fig. 2. Data objects (filled circles) and edges of an MST_{ext} computed over the transformed space of mutual reachability distances with $m_{\text{pts}} = 3$ and Euclidean metric (solid lines). “Self edges” and edge weights are omitted for the sake of clarity.

ALGORITHM 1: HDBSCAN* main steps

```
1. Compute the core distance w.r.t. $m_{pts}$ for all data objects in $\mathbf{X}$.
2. Compute an MST of $G_{m_{pts}}$, the Mutual Reachability Graph.
3. Extend the MST to obtain $MST_{ext}$, by adding for each vertex a “self edge” with the core
distance of the corresponding object as weight.
4. Extract the HDBSCAN* hierarchy as a dendrogram from $MST_{ext}$:
    4.1 For the root of the tree assign all objects the same label (single “cluster”).
    4.2 Iteratively remove all edges from $MST_{ext}$ in decreasing order of weights
        (in case of ties, edges must be removed simultaneously):
        4.2.1 Before each removal, set the dendrogram scale value of the current
              hierarchical level as the weight of the edge(s) to be removed.
        4.2.2 After each removal, assign labels to the connected component(s) that contain(s)
              the end vertex(-ices) of the removed edge(s), to obtain the next hierarchical level:
              assign a new cluster label to a component if it still has at least one edge, else assign
              it a null label (“noise”).
```

The idea given earlier can be applied to substantially simplify the HDBSCAN* hierarchy by focusing only on those hierarchical levels in which new clusters arise by a “true” split of an existing cluster, or in which clusters completely disappear, representing the levels in which the most significant changes in the clustering structure occur. To establish what constitutes a “true” split of a cluster we make use of the fact that noise objects are not considered to be clusters in HDBSCAN*, so their removal from a given cluster should not be considered as a split of that cluster, but just the removal of objects that are no longer connected to it at the corresponding density threshold. In other words, when decreasing ε, the ordinary removal of noise objects from a cluster means that the cluster has shrunk only, so the remaining objects should keep the same label.

Let us consider a simple illustrative example involving the toy dataset in Figure 2. The application of HDBSCAN* to those data with $m_{\text{pts}} = 3$ and Euclidean distance results in the hierarchy shown in Table I. The relevant hierarchical levels in this case are those corresponding to $\varepsilon = 7.1$, below which cluster $\mathbf{C}_1$ is split into $\mathbf{C}_2$ and $\mathbf{C}_3$, $\varepsilon = 3.04$, below which cluster $\mathbf{C}_2$ is split into $\mathbf{C}_4$ and $\mathbf{C}_5$, and $\varepsilon = 1.28$, 1.22, and 0.67, below which clusters $\mathbf{C}_3$, $\mathbf{C}_4$, and $\mathbf{C}_5$ disappear, respectively. In the remaining levels, clusters only shrink by losing objects that become noise. In the case of clusters $\mathbf{C}_3$

Table I. HDBSCAN* Hierarchy for the dataset in Figure 2, with $m_{\text{pts}} = 3$. Higher (lower) Hierarchical Levels are on the left (right). Values of ε in the Bottom row are those Assigned in Step 4.2.1 of Algorithm 1. The Remaining Values are the labels Assigned in Steps 4.1 and 4.2.2: a non-null Value i in the jth row Means that Object $\mathbf{x}_j$ Belongs to Cluster $\mathbf{C}_i$ at the Corresponding Level, Whereas a null Value Denotes Noise

$\mathbf{x}_1$	1	2	2	4	4	4	4	0	0	0	0
$\mathbf{x}_2$	1	2	2	4	4	4	4	4	4	0	0
$\mathbf{x}_3$	1	2	2	4	4	4	4	0	0	0	0
$\mathbf{x}_4$	1	2	2	0	0	0	0	0	0	0	0
$\mathbf{x}_5$	1	2	2	5	5	5	0	0	0	0	0
$\mathbf{x}_6$	1	2	2	5	5	5	5	5	5	5	0
$\mathbf{x}_7$	1	2	2	5	5	5	5	5	5	5	0
$\mathbf{x}_8$	1	2	2	5	5	5	5	5	5	5	0
$\mathbf{x}_9$	1	2	2	5	5	5	0	0	0	0	0
$\mathbf{x}_{10}$	1	2	0	0	0	0	0	0	0	0	0
$\mathbf{x}_{11}$	1	3	3	3	0	0	0	0	0	0	0
$\mathbf{x}_{12}$	1	3	3	3	3	0	0	0	0	0	0
$\mathbf{x}_{13}$	1	3	3	3	3	3	3	3	0	0	0
$\mathbf{x}_{14}$	1	3	3	3	3	0	0	0	0	0	0
ε	7.1	6.51	3.04	1.7	1.6	1.4	1.3	1.28	1.22	0.67	0

and $\mathbf{C}_4$, note that they are reduced to single objects in the intervals $\varepsilon = [1.28\ \ 1.6)$ and $\varepsilon = [1.22\ \ 1.3)$, respectively, before they completely disappear. This information is captured by the self-loops introduced in the extended MST, which describe the fact that $\mathbf{x}_{13}$ and $\mathbf{x}_2$ become isolated, yet dense objects in the corresponding intervals.

ALGORITHM 2: HDBSCAN* Step 4.2.2 with (optional) parameter $m_{clSize} \geq 1$

```
4.2.2 After each removal (to obtain the next hierarchical level), process one at a time each
cluster that contained the edge(s) just removed, by relabeling its resulting connected
subcomponent(s):
    Label spurious subcomponents as noise by assigning them the null label. If all
    subcomponents of a cluster are spurious, then the cluster has disappeared.
    Else, if a single subcomponent of a cluster is not spurious, keep its original cluster label
    (cluster has just shrunk).
    Else, if two or more subcomponents of a cluster are not spurious, assign new cluster
    labels to each of them ("true" cluster split).
```

The extended dendrogram that represents the density-based hierarchy in Table I is illustrated in Figure 3. Noise is indicated by thinner, red lines. The corresponding simplified cluster tree is displayed in Figure 4 (the values between parentheses in the nodes of the tree will be explained later). It is worth noticing that there are only five significant clusters in the tree, in contrast to 27 clusters that would exist in traditional hierarchical clusterings.

The idea for hierarchy simplification can be generalized by setting a minimum cluster size, a commonly used practice in real cluster analysis. In fact, in many practical applications of clustering (not just density-based clustering) a minimum cluster size is used as a user-specified parameter in order to prevent algorithms from finding very small clusters of objects that may be highly similar to each other just by chance, that is, as a consequence of the natural randomness associated with the use of a finite data sample (see, e.g., the notion of a *particle* in the work of Gupta et al. [2010]). Requiring a minimum cluster size, $m_{clSize} \geq 1$, allows only clusters with at least m_{clSize} objects to be reported, and the case in which a component with fewer than m_{clSize} objects is disconnected from a cluster should not be considered as a "true" split. We can adapt HDBSCAN* accordingly by changing Step 4.2.2 of Algorithm 1, as shown

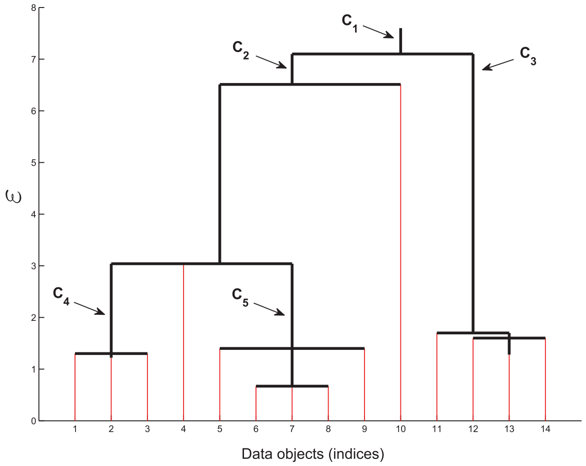


Fig. 3. Dendrogram corresponding to the HDBSCAN* hierarchy for the dataset in Figure 2, with $m_{\text{pts}} = 3$ (hierarchy in Table I). Thinner, red lines denote noise.

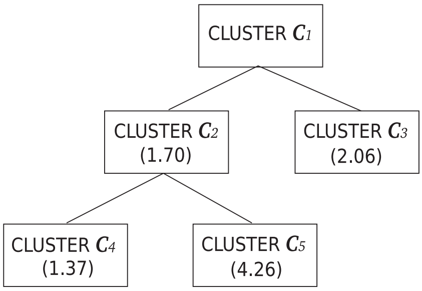


Fig. 4. Cluster tree corresponding to the hierarchy in Table I. Numerical values within parentheses stand for the stability of the clusters according to their relative excess of mass.

in Algorithm 2: a connected component is deemed *spurious* if it has fewer than m_{clSize} objects or, for $m_{clSize} = 1$, if it is an isolated, non-dense object (a vertex with no edges). Any spurious component is labeled as noise. Cardinality check and labeling of the components can be trivially performed by a graph traversal procedure starting from the end vertices of the removed edge(s). In practice, this simplification procedure can reduce dramatically the number of clusters in the hierarchy.

Note that the optional use of the parameter m_{clSize} represents an additional, independent control of the smoothing of the resulting cluster tree as a density estimate, in addition to the parameter m_{pts}. Note also that in the original definition of a density-based cluster for DBSCAN Ester et al. [1996], m_{pts} simultaneously acts as a direct control of the minimum cluster size, since *border* objects (which we do not consider in HDBSCAN*) belong to the same clusters as their corresponding core objects; consequently, the resulting clusters have no fewer than m_{pts} objects.[5]

To make HDBSCAN* more similar to previous density-based approaches like DBSCAN in this respect, and also to simplify its use, we can set $m_{clSize} = m_{\text{pts}}$, which turns m_{pts} into a single parameter that acts at the same time as a smoothing factor of the density estimates and an explicit threshold for the minimum size of clusters.

3.4. Computational Complexity

The asymptotic complexity of HDBSCAN* in Algorithms 1 and 2 is discussed in this section w.r.t. running time and memory space, considering two different scenarios. In the first scenario, the dataset $\mathbf{X}$ is available as an input to the algorithm. In the second (relational) scenario, the distance matrix D is available instead. It is important to remark that one has $m_{\text{pts}} \ll n$ (where $n = |\mathbf{X}|$) in any realistic application, so this assumption is made here.

Let us first consider the first scenario, in which $\mathbf{X}$ is available. We assume that the distance $d(\cdot, \cdot)$ between any pair of objects can be computed in $O(a)$ time—where a

[5] The DBSCAN algorithm, however, does not implement the possible multiple labels for border-objects but assigns a border object arbitrarily to just one of the clusters it belongs to.

is the number of attributes describing the objects—as usual for many dissimilarity functions. Then, in the most general case, Step 1 of the algorithm takes $O(a\,n^2)$ time, as it demands n K-NN queries (with $K = m_{\text{pts}}$), one per each data object. By using a simple implementation of Prim's algorithm based on an ordinary list search (instead of a heap), it is possible to construct the MST in Step 2 of HDBSCAN* in $O(n^2 + m)$ time, where m is the number of edges of the mutual reachability graph (with n vertices). In the present context, $m = n(n-1)/2$ (complete undirected graph), so Step 2 runs in $O(n^2 + m) \rightarrow O(n^2)$ time. Notice that it is *not* necessary to explicitly construct the mutual reachability graph, as the mutual reachability distances (edge weights) can be computed on demand. In this case, the MST can be computed in $O(a\,n^2)$ time. The number of edges in the MST is $n-1$, plus n additional "self-edges" in the extended MST (Step 3). These $2n-1$ edges need to be sorted so that Step 4 can be performed, which can be done in $O(n \log n)$. After the sorting procedure, Step 4 reduces to a series of relabelings of smaller and smaller subcomponents of the MST. Notice that at most two subcomponents must be relabeled after each edge removal, and that *spurious* components are never relabeled after they have become spurious. In the worst case, though, the whole relabeling procedure can still take $O(n^2)$ time. In total, the overall time complexity of the algorithm is thus $O(a\,n^2)$.

Still considering the first scenario, but now in terms of main memory requirements, it follows that one needs $O(a\,n)$ space to store the dataset $\mathbf{X}$ and $O(n)$ space to store the core distances in Step 1 of the algorithm. In Steps 2 and 3, recall that the mutual reachability graph does not need to be explicitly computed, and only the edges of the resulting MST_{ext} must be stored, which requires $O(n)$ space. During the execution of Step 4, only the hierarchical level being currently processed is needed at any point in time, which requires $O(n)$ space. Therefore, the overall space complexity of the algorithm is $O(a\,n)$.

In the case in which the distance matrix D is given *instead* of the dataset $\mathbf{X}$, the only change in terms of running time is that, as one can promptly access any distance $d(\cdot,\cdot)$ from D in constant time, the computations no longer depend on the dimension a of the data space, and therefore the time complexity reduces to $O(n^2)$. On the other hand, this requires that matrix D be stored in main memory, which results in $O(n^2)$ complexity in terms of space.

It is worth noticing that the running times discussed earlier can be reduced provided that some particular assumptions hold true, especially if the average case is considered. For instance, Step 1 of the algorithm can be computed in sub-quadratic time by using appropriate indexing structures for K-NN search in data spaces of low or moderate dimensionality. Even the MST in Step 2 might be computed (at least partially, as a forest) in sub-quadratic time, if an appropriate upper bound is imposed on the radius ε, so that the mutual reachability graph becomes sparse (as in OPTICS [Ankerst et al. 1999]).

In terms of space, if one wants to keep the whole hierarchy in main memory (instead of iteratively saving each hierarchical level on disk as they are computed), the hierarchy does not need to be kept in its complete form illustrated in Table I. All the information one needs to recover the complete hierarchy is, for each data object, a list (set) of all cluster labels associated with that object along the hierarchy and the corresponding hierarchical levels (scales) at which the object first belongs to each of those clusters. Typically, all these lists together will be much smaller than a matrix as illustrated in Table I, specially for larger values of m_{pts} and m_{clSize}.

Another alternative to optimize space is to store a compacted hierarchy containing only the most important hierarchical levels, namely, those in which the clusters in the cluster tree first appear or completely disappear. This can reduce significantly the size of the output file. A Java implementation of the algorithm that optionally supports this

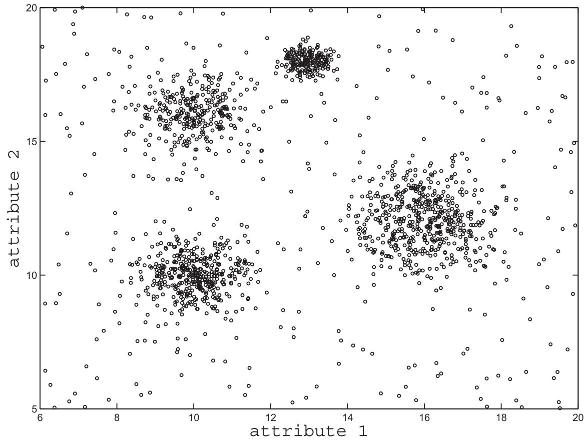


Fig. 5. Illustrative dataset with four clusters and background noise.

compaction strategy is available on request. As an example of this code's performance, for a dataset with 50,000 objects distributed in 50 clusters in a 50 dimensional Euclidean space, the algorithm, running on a domestic laptop[6], finished in 5.42 minutes and produced an output file of 14MB (containing the compacted hierarchy and the cluster tree).

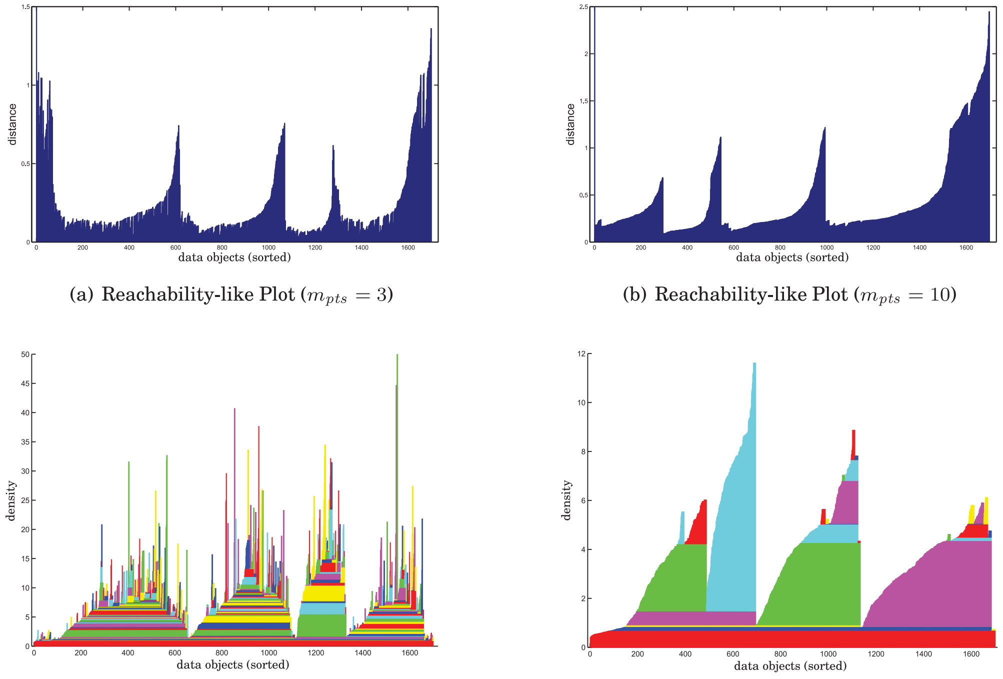
4. VISUALIZATION

The HDBSCAN* hierarchy can be visualized in different ways, for example, as a dendrogram or simplified cluster tree, as illustrated in Figures 3 and 4, respectively. However, while these plots can usually give a clear view of the cluster structure for small datasets, they may not be easy to read in more general application scenarios. Therefore, alternative forms of visualization may be needed.

Ankerst et al. [1999] proposed the well-known OPTICS algorithm whose primary output is the so-called *reachability plot*, which, roughly speaking, is a bar plot showing clusters as "dents" characterized by valleys and peaks along the bars. Sander et al. [2003] have shown that dendrograms in general (not necessarily density based) can be converted into, and visualized as, reachability-like plots. Provided that the data objects have already been sorted in the way they would be displayed in the dendrogram (i.e., with the objects in each cluster always placed next to each other), the basic idea is to plot a bar for each object whose height is the smallest dendrogram scale value at which the object gets merged into the same cluster as any of the preceding objects in the plot. For the HDBSCAN* hierarchy, an appropriate ordering in which objects in the same cluster necessarily appear next to each other when building the plot can be achieved by recursively sorting the subsets of objects in each cluster top-down the hierarchy, which is equivalent to a top-down level-wise lexicographic sorting according to the cluster labels.

As an illustrative example, let us consider the dataset in Figure 5, in which there are four clusters following 2D normal distributions and uniform background noise. The reachability-like plots corresponding to the HDBSCAN* hierarchies for these data with $m_{\text{pts}} = 3$ and $m_{\text{pts}} = 10$ are displayed in Figures 6(a) and 6(b), respectively. In both figures, the four natural clusters appear clearly as the most prominent "dents"

[6] MacBook Pro, 2.5GHz Intel Core i5, 8GB RAM, Mac OS X Lion 10.7.5, running Java 7 with algorithm settings set to $m_{\text{pts}} = m_{clSize} = 50$.



(a) Reachability-like Plot ($m_{pts} = 3$)

(b) Reachability-like Plot ($m_{pts} = 10$)

(c) Silhouette Plot ($m_{pts} = 3$)

(d) Silhouette Plot ($m_{pts} = 10$)

Fig. 6. Reachability and silhouette plots of the HDBSCAN* hierarchy for the dataset in Figure 5, for $m_{\text{pts}} = 3$ and $m_{\text{pts}} = 10$ ($m_{clsize} = m_{\text{pts}}$). Note that, since the clustering hierarchies are different for different m_{pts}, the sorting of the objects on the left- and right-hand figures are different.

in the plots, but for $m_{\text{pts}} = 10$ the plot is smoother and subclusters do not show up as apparently as for $m_{\text{pts}} = 3$.

For density-based hierarchies, an alternative yet related form of visualization consists in sorting the data objects in the same way as in the reachability plots, but setting the height of each bar in the plot to the highest density such that the corresponding data object is still part of a cluster (not yet noise). This is in essence a type of dimensionless representation of a silhouette plot of densities [Muller and Sawitzki 1991]. Such a silhouette plot can be made more sophisticated, as proposed by Gupta et al. [2010], by plotting in different colors the different segments of the bars that correspond to the density intervals along which an object belongs to different clusters in the hierarchy (each subcluster is assigned a particular color that is guaranteed to be different from its parent's color). For the dataset in Figure 5, the silhouette plots of the HDBSCAN* hierarchies are illustrated in Figures 6(c) and 6(d). For $m_{\text{pts}} = 3$ the plot is spikier and, from the changes in colors, it is clear that the hierarchy is more complex (many subclusters in the cluster tree). For $m_{\text{pts}} = 10$ the plot is smoother and we can see that the number of subclusters is drastically reduced.

Reachability or silhouette plots display hierarchical clusterings in different ways, but we can see from the example in Figure 6 that they both not only allow a visualization of the structure of clusters found by the algorithm, but also the effects of different settings of the algorithm on the clustering results. As such, these plots can be useful auxiliary tools for exploratory data analysis, particularly in the domain of unsupervised learning by means of the HDBSCAN* framework.

5. OPTIMAL NONHIERARCHICAL CLUSTERING

While hierarchical clustering is an important and widely used tool for exploratory data analysis, there are also many important applications in which a user is interested in extracting from a hierarchical clustering a *flat* or *partition-like* solution consisting of the most prominent, nonoverlapping clusters. Choosing one of the available hierarchical levels as such a solution is the well-known problem of performing a horizontal cut through a dendrogram [Jain and Dubes 1988]. However, for density-based methods, a horizontal cut corresponds to a single, global density threshold which may not simultaneously detect clusters having largely varying local densities [Ankerst et al. 1999; Tan et al. 2006; Kriegel et al. 2011]. In this section, we describe an algorithm that provides the optimal solution to the formal optimization problem of maximizing the overall quality of the set of clusters extracted from *local* cuts through the HDBSCAN* hierarchy, in both an unsupervised, as well as a semisupervised setting.

5.1. Cluster Stability

Without loss of generality, let us initially consider that the data objects are described by a single continuous-valued attribute x. Recall from the introduction that, following Hartigan's model [Hartigan 1975], the density-contour clusters of a given density $f(x)$ on $\Re$ at a given density level λ are the maximal connected subsets of the level set defined as $\{x \mid f(x) \geq \lambda\}$. From Section 2, it follows that DBSCAN* estimates density-contour clusters using a density threshold $\lambda = 1/\varepsilon$ and a non-normalized K-NN estimate (for $K = m_{\text{pts}}$) of $f(x)$, given by $1/d_{\text{core}}(x)$.[7]

HDBSCAN* produces all possible DBSCAN* solutions w.r.t. a given value of m_{pts} and all thresholds $\lambda = 1/\varepsilon$ in $[0, \infty)$. Intuitively, when increasing λ (i.e., decreasing ε), clusters get smaller and smaller, until they disappear or break into subclusters. This observation gives rise to the intuition that, roughly speaking, more prominent clusters tend to "survive" longer after they appear, and which is essentially the rationale behind the definition of *cluster lifetime* from classic hierarchical cluster analysis [Jain and Dubes 1988; Fred and Jain 2005]. The lifetime of a given cluster in a traditional dendrogram is defined as the length of the dendrogram scale along those hierarchical levels in which the cluster exists. In traditional dendrograms, however, the removal of a single object suffices to characterize the dissolution of a cluster and appearance of new ones, which makes the original definition inappropriate in the density-based context. Therefore, a different measure of stability is needed that considers the dissolution of a cluster under the broader perspective discussed in Section 3.3, which accounts for the presence of noise and spurious components along the hierarchy. Such a measure should also take into account the individual density profiles of the objects belonging to a cluster, that is, their possibly different *lifetimes* as members of that cluster (before they become noise or members of another cluster).

To formalize the aforementioned idea, we adapt the notion of *excess of mass* [Muller and Sawitzki 1991], first introduced by Hartigan [1987] and more recently used in the context of estimates of level sets for continuous-valued PDF [Stuetzle and Nugent 2010]. Imagine increasing the density level λ and assume that a density-contour cluster $\mathbf{C}_i$ appears at level $\lambda_{\min}(\mathbf{C}_i)$, by definition as a maximal connected subset of the level set $\{x \mid f(x) \geq \lambda_{\min}(\mathbf{C}_i)\}$. The excess of mass of $\mathbf{C}_i$ is defined in Equation (1), and illustrated in Figure 7, in which the darker shaded areas represent the excesses of mass of three

[7] From this perspective, the role of m_{pts} as a classic smoothing factor for such an estimate becomes clearer. It is worth remarking, however, that DBSCAN* does not attempt to produce a very accurate estimate of the true PDF of the data. The density estimate is used essentially to discriminate between noise and non-noise data objects, which contributes to making m_{pts} a particularly noncritical parameter.

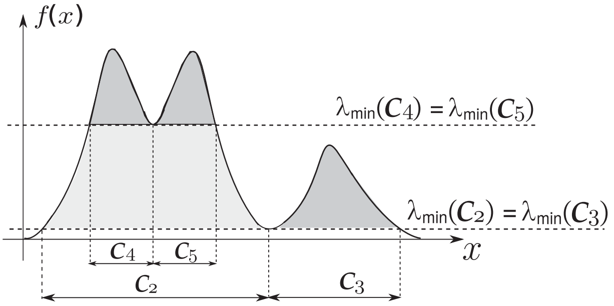


Fig. 7. Illustration of a density function, clusters, and excesses of mass.

clusters, $\mathbf{C}_3$, $\mathbf{C}_4$, and $\mathbf{C}_5$. The excess of mass of $\mathbf{C}_2$ (not highlighted in the figure) encompasses those of its children $\mathbf{C}_4$ and $\mathbf{C}_5$.

$$E(\mathbf{C}_i) = \int_{x \in \mathbf{C}_i} (f(x) - \lambda_{\min}(\mathbf{C}_i))\, dx \tag{1}$$

Since the excess of mass of a cluster necessarily embodies those of all its descendants, this measure exhibits a monotonic behavior along any branch of the cluster tree. As a consequence, it cannot be used to compare nested clusters, such as $\mathbf{C}_2$ against $\mathbf{C}_4$ and $\mathbf{C}_5$. To be able to do so, we introduce the notion of *relative excess of mass* of a cluster $\mathbf{C}_i$, which appears at level $\lambda_{\min}(\mathbf{C}_i)$, as:

$$E_R(\mathbf{C}_i) = \int_{x \in \mathbf{C}_i} (\lambda_{\max}(x, \mathbf{C}_i) - \lambda_{\min}(\mathbf{C}_i))\, dx, \tag{2}$$

where $\lambda_{\max}(x, \mathbf{C}_i) = \min\{f(x), \lambda_{\max}(\mathbf{C}_i)\}$ and $\lambda_{\max}(\mathbf{C}_i)$ is the density level at which $\mathbf{C}_i$ is split or disappears. For example, for cluster $\mathbf{C}_2$ in Figure 7 it follows that $\lambda_{\max}(\mathbf{C}_2) = \lambda_{\min}(\mathbf{C}_4) = \lambda_{\min}(\mathbf{C}_5)$. The corresponding relative excess of mass is represented by the lighter shaded area in Figure 7.

For a HDBSCAN* hierarchy, in which we have a finite dataset $\mathbf{X}$, cluster labels, and density thresholds associated with each hierarchical level, we can adapt Equation (2) to define the *stability* of a cluster $\mathbf{C}_i$ as:

$$S(\mathbf{C}_i) = \sum_{\mathbf{x}_j \in \mathbf{C}_i} \left(\lambda_{\max}(\mathbf{x}_j, \mathbf{C}_i) - \lambda_{\min}(\mathbf{C}_i)\right) = \sum_{\mathbf{x}_j \in \mathbf{C}_i} \left(\frac{1}{\varepsilon_{\min}(\mathbf{x}_j, \mathbf{C}_i)} - \frac{1}{\varepsilon_{\max}(\mathbf{C}_i)}\right) \tag{3}$$

where $\lambda_{\min}(\mathbf{C}_i)$ is the minimum density level at which $\mathbf{C}_i$ exists, $\lambda_{\max}(\mathbf{x}_j, \mathbf{C}_i)$ is the density level beyond which object $\mathbf{x}_j$ no longer belongs to cluster $\mathbf{C}_i$, and $\varepsilon_{\max}(\mathbf{C}_i)$ and $\varepsilon_{\min}(\mathbf{x}_j, \mathbf{C}_i)$ are the corresponding values for the threshold ε.

Recalling the illustrative example described in Section 3.3, it follows that object $\mathbf{x}_{11}$, for instance, belongs to cluster $\mathbf{C}_3$ in the radius interval $\varepsilon = [1.7\ \ 7.1)$, that is, in the density interval $\lambda = (1/7.1\ \ 1/1.7] = (0.14\ \ 0.59]$. The first value of this density interval is precisely $\lambda_{\min}(\mathbf{C}_3)$, which is common to all objects of that cluster, namely, $\mathbf{x}_{11}$, $\mathbf{x}_{12}$, $\mathbf{x}_{13}$, and $\mathbf{x}_{14}$. The second value is $\lambda_{\max}(\mathbf{x}_{11}, \mathbf{C}_3) = 0.59$, above which $\mathbf{x}_{11}$ no longer belongs to cluster $\mathbf{C}_3$. Analogously, it follows that $\lambda_{\max}(\mathbf{x}_{12}, \mathbf{C}_3) = \lambda_{\max}(\mathbf{x}_{14}, \mathbf{C}_3) = 1/1.6 = 0.625$ and $\lambda_{\max}(\mathbf{x}_{13}, \mathbf{C}_3) = 1/1.28 = 0.781$. Thence, the relative excess of mass of cluster $\mathbf{C}_3$ according to Equation (3) is $S(\mathbf{C}_3) = 2.06$. The relative excesses of mass for the other clusters are displayed in the nodes of the cluster tree in Figure 4. The value for the root $\mathbf{C}_1$ is not shown to emphasize that $\mathbf{C}_1$, which represents the whole dataset, is not deemed a valid cluster in the cluster extraction algorithm to be described next.

Before discussing the extraction algorithm, however, it is important to remark that the cluster stabilities as described earlier can be simultaneously computed during the construction of the HDBSCAN* hierarchy, with very little additional processing and without affecting the asymptotic complexity of the algorithm. To do so, all one needs is, for each object $\mathbf{x}_j$, to keep track of the changes in $\mathbf{x}_j$'s label so that the values $\lambda_{\max}(\mathbf{x}_j, \mathbf{C}_i) - \lambda_{\min}(\mathbf{C}_i)$ can be computed and accumulated for each cluster $\mathbf{C}_i$ to which object $\mathbf{x}_j$ will belong. The additional memory required is $O(n + \kappa)$, where n is the number of objects and κ is the number of clusters in the cluster tree. Since κ cannot be greater than $2n - 1$ (actually, one typically has $\kappa \ll n$ when using the hierarchy simplification procedure described in Section 3.3), the additional memory required is $O(n)$, which therefore does not change the asymptotic complexity of HDBSCAN* in terms of space either.

5.2. Optimization Algorithm for Unsupervised Cluster Extraction

Let $\{\mathbf{C}_2, \ldots, \mathbf{C}_\kappa\}$ be the collection of all clusters in the simplified cluster hierarchy (tree) generated by HDBSCAN*, except the root $\mathbf{C}_1$, and let $S(\mathbf{C}_i)$ denote the stability value of each cluster. The goal is to extract the most "prominent" clusters (plus possibly noise) as a flat, nonoverlapping clustering solution. This task can be formulated as an optimization problem with the objective of maximizing the overall aggregated stabilities of the extracted clusters, in the following way:

$$\max_{\delta_2, \ldots, \delta_\kappa} \quad J = \sum_{i=2}^{\kappa} \delta_i \, S(\mathbf{C}_i)$$
$$\text{subject to} \begin{cases} \delta_i \in \{0, 1\}, \quad i = 2, \ldots, \kappa \\ \text{exactly one } \delta_{(\cdot)} = 1 \text{ in each path from a leaf cluster to the root} \end{cases} \tag{4}$$

where δ_i indicates whether cluster $\mathbf{C}_i$ is included into the flat solution ($\delta_i = 1$) or not ($\delta_i = 0$). The constraints imposed on the decision variables $\delta_2, \ldots, \delta_\kappa$ prevent clusters on the same branch of the tree to be simultaneously selected; clearly, for inclusion in a flat solution, nested clusters must be mutually exclusive, that is, each object can only be assigned a single label.

To solve Problem (4) both globally and efficiently, we notice that the subselection of clusters in any subtree of the cluster tree represents a subproblem of the very same nature of the original problem (i.e., the one that refers to the complete tree). From this observation, a dynamic programming strategy can be applied that incrementally solves subproblems (subtrees) of increasing sizes, starting from the leaves and aggregating the intermediate solutions upwards in the tree. Specifically, we process every node except the root, starting from the leaves (bottom-up), deciding at each node $\mathbf{C}_i$ whether $\mathbf{C}_i$ or the best-so-far selection of clusters in $\mathbf{C}_i$'s subtrees should be selected. To be able to make this decision locally at $\mathbf{C}_i$, we propagate and update the total stability $\hat{S}(\mathbf{C}_i)$ of clusters selected in the subtree rooted at $\mathbf{C}_i$ in the following, recursive way:

$$\hat{S}(\mathbf{C}_i) = \begin{cases} S(\mathbf{C}_i), & \text{if } \mathbf{C}_i \text{ is a leaf node} \\ \max\{S(\mathbf{C}_i), \hat{S}(\mathbf{C}_{i_l}) + \hat{S}(\mathbf{C}_{i_r})\} & \text{if } \mathbf{C}_i \text{ is an internal node} \end{cases} \tag{5}$$

where $\mathbf{C}_{i_l}$ and $\mathbf{C}_{i_r}$ are the left and right children of $\mathbf{C}_i$ (for the sake of simplicity, we discuss the case of binary trees; the generalization to n-ary trees is trivial).

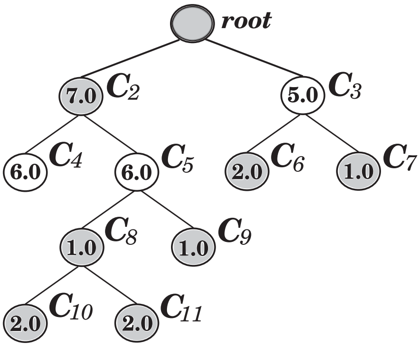


Fig. 8. Illustration of the optimal selection of clusters from a given cluster tree.

Algorithm 3 gives the pseudocode for finding the optimal solution to Problem (4). Figure 8 illustrates the algorithm. Clusters $\mathbf{C}_{10}$ and $\mathbf{C}_{11}$ together are better than $\mathbf{C}_8$, which is then discarded. However, when the set $\{\mathbf{C}_{10}, \mathbf{C}_{11}, \mathbf{C}_9\}$ is compared to $\mathbf{C}_5$, they are discarded as $\mathbf{C}_5$ is better. Clusters $\{\mathbf{C}_4\}$ and $\{\mathbf{C}_5\}$ are better than $\mathbf{C}_2$, and $\mathbf{C}_3$ is better than $\{\mathbf{C}_6, \mathbf{C}_7\}$, so that in the end, only clusters $\mathbf{C}_3$, $\mathbf{C}_4$, and $\mathbf{C}_5$ remain, which is the optimal solution to Problem (4) with $J = 17$.

ALGORITHM 3: Solution to Problem (4)

```
1. Initialize $\delta_2 = \cdots = \delta_\kappa = 1$, and, for all leaf nodes, set $\hat{S}(\mathbf{C}_h) = S(\mathbf{C}_h)$.
2. Starting from the deepest levels, do bottom-up (except for the root):
    2.1 If $S(\mathbf{C}_i) < \hat{S}(\mathbf{C}_{i_l}) + \hat{S}(\mathbf{C}_{i_r})$, set $\hat{S}(\mathbf{C}_i) = \hat{S}(\mathbf{C}_{i_l}) + \hat{S}(\mathbf{C}_{i_r})$ and set $\delta_i = 0$.
    2.2 Else: set $\hat{S}(\mathbf{C}_i) = S(\mathbf{C}_i)$ and set $\delta_{(\cdot)} = 0$ for all clusters in $\mathbf{C}_i$'s subtrees.
```

Notice that Step 2.2 of Algorithm 3 can be implemented in a more efficient way by not setting $\delta_{(.)}$ values to 0 for discarded clusters down in the subtrees (which could happen multiple times for the same cluster). Instead, in a simple postprocessing procedure, the tree can be traversed top-down in order to find, for each branch, the shallowest cluster that has not been discarded ($\delta_{(\cdot)} = 1$). Thus, Algorithm 3 can be implemented with two traversals through the tree, one bottom-up and another one top-down. This results in an asymptotic complexity of $O(\kappa)$, both in terms of running time and memory space, where κ is the number of clusters in the simplified cluster tree, which is $O(n)$ in the worst case.

5.3. Semisupervised Cluster Extraction

In many different application scenarios, a certain, usually small amount of information about the data may be available that allows one to perform clustering in a semisupervised rather than in a completely unsupervised way. The most common type of information appears in the form of instance-level constraints, which encourage pairs of objects to be grouped together in the same cluster (*should-link constraints*) or to be separated apart in different clusters (*should-not-link constraints*) [Wagstaff 2002; Basu et al. 2008]. These are *external*, *explicit* constraints provided by the user or analyst in *addition* to the *internal*, *implicit* constraints that follow from the inductive bias of the clustering model that has been adopted.

HDBSCAN* is peculiar in the sense that it produces a complete, exhaustive collection of candidate clusters that satisfy the *internal* constraints embedded into the corresponding clustering model, namely, the requirements for a density-based cluster according to Definition 2.4 (given a value of m_{pts}). This allows semisupervision to take place as a postprocessing stage of HDBSCAN*, aimed at selecting those clusters that altogether better satisfy the external constraints provided by the user, while not violating the original grounds of the underlying clustering model. This approach can be interpreted as the use of external constraints as *preferences* rather than hard requirements, which is in conformity with the fact that such constraints typically represent a user's *preliminary expectation* about the application domain (without knowing the actual structure of clusters inherent in the data). By using this approach, we do not need to modify the clustering algorithm itself, we need instead only to adapt the cluster selection mechanism, which is naturally less susceptible to overfitting.

Adapting the cluster selection mechanism described in Section 5.2 to deal with instance-level constraints is rather straightforward by noticing that we can replace the values of stability for each cluster, $S(\mathbf{C}_i)$, with the fraction of constraints that are satisfied by that cluster, $\Gamma(\mathbf{C}_i) = \frac{1}{2n_c}\sum_{\mathbf{x}_j \in \mathbf{C}_i} \gamma(\mathbf{x}_j, \mathbf{C}_i)$, where n_c is the total number of available constraints and $\gamma(\mathbf{x}_j, \mathbf{C}_i)$ is the number of constraints *involving object* $\mathbf{x}_j$ that are satisfied (not violated) if $\mathbf{C}_i$ is selected as part of the final flat solution. The scaling constant $^1/_2$ in $\Gamma(\mathbf{C}_i)$ is due to the fact that a single constraint involves a pair of objects and, as such, it is taken into account twice in the sum. Term $\Gamma(\mathbf{C}_i)$ is zero for clusters whose objects are not involved in any constraints.

Merely replacing $S(\mathbf{C}_i)$ with $\Gamma(\mathbf{C}_i)$, however, would not account for the fact that some objects may not belong to any of the selected clusters, that is, they may be labeled as noise in the final flat solution. For example, let us recall the toy problem illustrated in Section 3.3. In particular, let us consider a candidate flat solution given by $\{\mathbf{C}_3, \mathbf{C}_4, \mathbf{C}_5\}$. From the hierarchy in Table I it can be seen that object $\mathbf{x}_4$, for instance, belongs to none of these clusters. It belongs to $\mathbf{C}_2$ and becomes noise at the same density level as this cluster is split into $\mathbf{C}_4$ and $\mathbf{C}_5$. Something similar is observed w.r.t. $\mathbf{x}_{10}$. Objects $\mathbf{x}_4$ and $\mathbf{x}_{10}$ would therefore be labeled as noise in a flat solution composed of clusters $\mathbf{C}_3$, $\mathbf{C}_4$, and $\mathbf{C}_5$. Notice that since a noise object is, by definition, not clustered with any other object, a should-not-link constraint involving one or both objects labeled as noise must be deemed satisfied, while a should-link constraint must be deemed violated. Noise objects, therefore, also affect the degree of satisfaction/violation of constraints, and thus they cannot be dismissed.

In order to account for the noise, we keep track of objects that are part of a cluster but not part of its subclusters in the hierarchy (if any). This information can be represented by means of *"virtual" nodes* in the simplified cluster tree. We assume that every internal node of the cluster tree has one such virtual child node, even though only some of them may actually be associated with noise objects. The virtual child node of a cluster $\mathbf{C}_l$ in the cluster tree will be denoted hereafter as $\mathbf{C}_l^{\emptyset}$. We then define $\Gamma(\mathbf{C}_l^{\emptyset}) = \frac{1}{2n_c}\sum_{\mathbf{x}_j \in \mathbf{C}_l^{\emptyset}} \gamma(\mathbf{x}_j, \mathbf{C}_l^{\emptyset})$ as the fraction of constraint satisfactions involving the noise objects associated with $\mathbf{C}_l^{\emptyset}$ (which is zero if no noise object is associated with $\mathbf{C}_l^{\emptyset}$).

For example, consider again the hierarchy in Table I and assume that the user has provided three should-link constraints, $(\mathbf{x}_1, \mathbf{x}_5)$, $(\mathbf{x}_3, \mathbf{x}_9)$, and $(\mathbf{x}_2, \mathbf{x}_6)$, and a single should-not-link constraint, $(\mathbf{x}_7, \mathbf{x}_{10})$. The internal nodes are $\mathbf{C}_1$ and $\mathbf{C}_2$. Since there is no noise object in the virtual node of the root, $\mathbf{C}_1^{\emptyset}$, it follows that $\Gamma(\mathbf{C}_1^{\emptyset}) = 0$. The noise objects in $\mathbf{C}_2^{\emptyset}$ are $\mathbf{x}_4$ and $\mathbf{x}_{10}$, but there is no constraint involving object $\mathbf{x}_4$, then $\gamma(\mathbf{x}_4, \mathbf{C}_2^{\emptyset}) = 0$. The only constraint involving object $\mathbf{x}_{10}$ is satisfied in $\mathbf{C}_2^{\emptyset}$, that is, $\gamma(\mathbf{x}_{10}, \mathbf{C}_2^{\emptyset}) = 1$ (because $\mathbf{x}_{10}$, as noise, is not grouped with $\mathbf{x}_7$). Therefore, $\Gamma(\mathbf{C}_2^{\emptyset}) = \frac{1}{2\times 4}(0+1) = \frac{1}{8}$. As for the

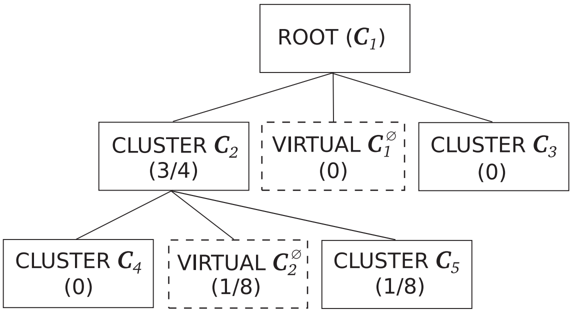


Fig. 9. Cluster tree corresponding to the hierarchy in Table I, augmented with virtual nodes. Numerical values within parentheses stand for the fractions of constraint satisfactions associated with each (original or virtual) node.

nonvirtual nodes, we have, for instance, $\Gamma(\mathbf{C}_5) = \frac{1}{8}\sum_{i=5}^{9}\gamma(\mathbf{x}_i, \mathbf{C}_5) = \frac{1}{8}(0+0+1+0+0) = \frac{1}{8}$. The values for the other clusters can be computed similarly as $\Gamma(\mathbf{C}_2) = \frac{6}{8}$, $\Gamma(\mathbf{C}_3) = 0$, and $\Gamma(\mathbf{C}_4) = 0$. The cluster tree is shown in Figure 9.

Once the values of $\Gamma(\cdot)$ have been computed for all (original and virtual) nodes of the tree, the algorithm described in Section 5.2 can be slightly modified so as to maximize a cost function $J_{SS} \in [0, 1]$ given by the fraction of constraints that are satisfied or, equivalently, the complement of the fraction of constraints that are violated in the final flat solution (such that maximizing J_{SS} is equivalent to minimizing the number of constraint violations). To do so, we need to redefine the recursion in Equation (5) as:

$$\hat{\Gamma}(\mathbf{C}_i) = \begin{cases} \Gamma(\mathbf{C}_i), & \text{if } \mathbf{C}_i \text{ is a nonvirtual leaf node} \\ \max\{\Gamma(\mathbf{C}_i), \hat{\Gamma}(\mathbf{C}_{i_l}) + \hat{\Gamma}(\mathbf{C}_{i_r}) + \Gamma(\mathbf{C}_i^{\emptyset})\}, & \\ \qquad\qquad\qquad \text{if } \mathbf{C}_i \text{ is an internal node} \end{cases} \tag{6}$$

and then replace $S(\cdot)$, $\hat{S}(\cdot)$, and "$\hat{S}(\mathbf{C}_{i_l}) + \hat{S}(\mathbf{C}_{i_r})$" in Algorithm 3 with $\Gamma(\cdot)$, $\hat{\Gamma}(\cdot)$, and "$\hat{\Gamma}(\mathbf{C}_{i_l}) + \hat{\Gamma}(\mathbf{C}_{i_r}) + \Gamma(\mathbf{C}_i^{\emptyset})$", respectively.

In the example of Figure 9, cluster $\mathbf{C}_3$ will necessarily be included into the optimal flat solution as it does not compete with any other descendant or ascendant candidate cluster. In the other branch of the root, the algorithm will compare cluster $\mathbf{C}_2$ (with $\frac{3}{4}$ of the whole set of constraints satisfied) against its descendants (with $0 + \frac{1}{8} + \frac{1}{8} = \frac{1}{4}$ of the set of constraints satisfied), and will select $\{\mathbf{C}_2, \mathbf{C}_3\}$ as the optimal solution, with $J_{SS}^* = 0 + \frac{3}{4} = \frac{3}{4}$ (i.e., $1 - J_{SS}^* = \frac{1}{4}$ violations). In contrast, the optimal *unsupervised* solution ($\{\mathbf{C}_3, \mathbf{C}_4, \mathbf{C}_5\}$, corresponding to the tree in Figure 4), would be suboptimal according to the semisupervised objective function, with $J_{SS} = \frac{1}{4}$ (i.e., $\frac{3}{4}$ violations).

In case of a tie in the local decision involving the subtree rooted at a cluster $\mathbf{C}_i$, that is, if $\hat{\Gamma}(\mathbf{C}_i) = \hat{\Gamma}(\mathbf{C}_{i_l}) + \hat{\Gamma}(\mathbf{C}_{i_r}) + \Gamma(\mathbf{C}_i^{\emptyset})$, which typically happens when the objects in such a subtree ($\mathbf{C}_i$ and its subclusters) are not involved in any constraints at all, the decision can still be made in an unsupervised way, based on cluster stability. This can be interpreted as using unsupervised stability as a secondary objective to resolve ties in the semisupervised scenario. These two objectives could also be combined into a single one by means of a convex combination [Campello et al. 2013b], but we omit the details here for the sake of compactness.

In terms of complexity, the only additional cost w.r.t. the unsupervised version of the algorithm (Section 5.2) refers to the need for computing the values of Γ and also for storing the virtual nodes. The number of virtual nodes is equal to the number of internal nodes, which cannot exceed $O(n)$, so their presence does not change the complexity of the method in terms of memory space. Computing the values of Γ for both the original and virtual nodes of the tree can be done with a single pass through the simplified clustering hierarchy, checking for each constraint the labels of the pair of objects involved in that constraint. The simplified hierarchy is typically much smaller than the complete hierarchy, but when the worst case is considered, the running time to compute Γ for all nodes is $O(n_c\, n)$. If the number of constraints, n_c, is at most proportional to the number of data objects, n, as usual in many practical applications, then the worst-case runtime complexity of the method is also the same as for the unsupervised case, that is, $O(n)$.

6. OUTLIER DETECTION

In the literature, an object's outlierness is typically assessed w.r.t. some prespecified reference set, such as a *local* neighborhood or the *global* database. On one hand, by considering the global database as a reference set, as in global outlier detection methods (e.g., K-NN-outlier [Ramaswamy et al. 2000]), outliers may be overlooked when they are unusual only w.r.t. other objects lying in the same local region of the data space. On the other hand, by considering a predetermined local neighborhood as a reference set, outliers can be missed for they may be compared only with other outliers, which can happen for a local outlier detection method (e.g., LOF [Breunig et al. 2000]). Such a *global versus local* dilemma has persisted over years in the unsupervised outlier detection literature, with an exception being the method LOCI [Papadimitriou et al. 2003], which can, to some extent, find global and local outliers if the sampling radius $r_{\max}$ for the neighborhood of points is set so large as to include the whole dataset. The solution to the dilemma is to use a "dynamic" (rather than predetermined) reference set of objects. Starting from the usual assumption that there are one or more mechanisms responsible for generating the objects deemed nonsuspicious, and noticing that clusters are natural candidates to model such mechanism(s), the scope of the reference set can be adjusted for each object based on the closest cluster (in a density-based perspective) within the density-based hierarchy. By doing so, the locally suitable reference set can actually be global for some objects while being strictly local for other objects.[8] This approach provides a much more meaningful reference set for each point than a spherical neighborhood (even if evaluated at different radii as in LOCI), in the Hawkins sense of an outlier.

Consider a data object $\mathbf{x}_i$ for which the degree of outlierness has to be determined. Its closest cluster from a density-based perspective is the cluster $\mathbf{C}_j$ for which $\mathbf{x}_i \in \mathbf{C}_j$ at the density level corresponding to the density of $\mathbf{x}_i$, that is, at density $f(\mathbf{x}_i) = 1/d_{\text{core}}(\mathbf{x}_i)$ (or, equivalently, at radius $\varepsilon = d_{core}(\mathbf{x}_i)$). Note that $\mathbf{C}_j$ as defined earlier is the first cluster/node to which $\mathbf{x}_i$ gets attached when this object switches from noise to core as the density threshold is decreased (radius is increased) bottom-up through the hierarchy. The basic intuition here is that the densest object in cluster $\mathbf{C}_j$, say $\mathbf{x}_l$, can be considered the most inlier object and used as a reference for the other objects that have $\mathbf{C}_j$ as their closest density-based cluster. Object $\mathbf{x}_l$ is the one that "survives" the longest before all objects of the cluster become noise as we increase the density threshold top-down through the hierarchy. Let us refer to the density of such a densest object $\mathbf{x}_l$ of a cluster $\mathbf{C}_j$, that is, $f(\mathbf{x}_l) = 1/d_{\text{core}}(\mathbf{x}_l)$, as the referential density $f_{\max}(\mathbf{x}_i)$ for every object $\mathbf{x}_i$ that has $\mathbf{C}_j$ as its closest density-based cluster. Then, we can define

[8] It will be global if an object's closest density-based cluster is the root of the cluster tree. For example, if the tree contains a single "cluster", which models a dataset with unimodal distribution (e.g., Gaussian), this will be necessarily the case for all data objects.

a measure of outlierness for object $\mathbf{x}_i$ as

$$\frac{f_{\max}(\mathbf{x}_i) - f(\mathbf{x}_i)}{f_{\max}(\mathbf{x}_i)} \tag{7}$$

where $f(\mathbf{x}_i) = 1/d_{\text{core}}(\mathbf{x}_i)$ and $f_{\max}(\mathbf{x}_i) = 1/d_{\text{core}}(\mathbf{x}_l)$, with $\mathbf{x}_l$ being the densest object of $\mathbf{x}_i$'s closest cluster in the density-based hierarchy. Note that for the objects in the densest area of a cluster, $f_{\max}(\mathbf{x}_i) - f(\mathbf{x}_i)$ tends to be close to zero, so Expression (7) also tends to be close to zero, which indicates inlier objects. For objects that are only loosely density-connected to a cluster, the numerator $f_{\max}(\mathbf{x}_i) - f(\mathbf{x}_i)$ tends to be large (but $\leq f_{\max}$) and, in the limit, Expression (7) tends to 1, indicating the maximum degree of outlierness.

For the sake of illustration, let us recall the example in Figure 2 and Table I. Considering, for instance, object $\mathbf{x}_{10}$, it can be readily seen from Table I that its closest density-based cluster is $\mathbf{C}_2$. In fact, the density of object $\mathbf{x}_{10}$ is $f(\mathbf{x}_{10}) = 1/6.51$, at which $\mathbf{x}_{10}$ belongs to $\mathbf{C}_2$ and above which $\mathbf{x}_{10}$ is noise. The density of the densest object in $\mathbf{C}_2$ is $f(\mathbf{x}_6) = f(\mathbf{x}_7) = f(\mathbf{x}_8) = 1/0.67$, then, by definition, $f_{\max}(\mathbf{x}_{10}) = 1/0.67$. Therefore, the degree of outlierness of object $\mathbf{x}_{10}$ according to Expression (7) is

$$\frac{1/0.67 - 1/6.51}{1/0.67} = 0.8971.$$

Now, if we consider objects $\mathbf{x}_6$, $\mathbf{x}_7$, and $\mathbf{x}_8$, it is clear from Table I that: (i) their closest density-based cluster is $\mathbf{C}_5$; and (ii) they are actually the densest objects in that cluster, with $f(\mathbf{x}_6) = f(\mathbf{x}_7) = f(\mathbf{x}_8) = 1/0.67$. Therefore, $f_{\max}(\mathbf{x}_6) = f_{\max}(\mathbf{x}_7) = f_{\max}(\mathbf{x}_8) = 1/0.67$ and, according to Expression (7), the degree of outlierness of all these three objects is zero.

Expression (7), however, does not account for the fact that, if a minimum cluster size greater than one is adopted ($m_{clSize} > 1$; see Section 3.3), the lowest density threshold below which an object $\mathbf{x}_i$ first gets attached to a cluster, say $\lambda(\mathbf{x}_i)$, and the highest density threshold above which all objects of such a cluster are labeled noise, say $\lambda_{\max}(\mathbf{x}_i)$, no longer necessarily correspond to the densities $f(\mathbf{x}_i)$ and $f_{\max}(\mathbf{x}_i)$, respectively. To take the most general case into account, we generalize Expression (7) and define our *Global-Local Outlier Score from Hierarchies* (GLOSH) as:

$$\text{GLOSH}(\mathbf{x}_i) = \frac{\lambda_{\max}(\mathbf{x}_i) - \lambda(\mathbf{x}_i)}{\lambda_{\max}(\mathbf{x}_i)} = \frac{\frac{1}{\varepsilon_{\max}(\mathbf{x}_i)} - \frac{1}{\varepsilon(\mathbf{x}_i)}}{\frac{1}{\varepsilon_{\max}(\mathbf{x}_i)}} = 1 - \frac{\varepsilon_{\max}(\mathbf{x}_i)}{\varepsilon(\mathbf{x}_i)} \tag{8}$$

where $\varepsilon_{\max}(\mathbf{x}_i)$ and $\varepsilon(\mathbf{x}_i)$ are the values for the radii ε associated with the density thresholds $\lambda_{\max}(\mathbf{x}_i)$ and $\lambda(\mathbf{x}_i)$, respectively.

In order to compute GLOSH in Equation (8), one needs only the first (last) cluster to which object $\mathbf{x}_i$ belongs bottom-up (top-down) through the hierarchy, the lowest radius at which $\mathbf{x}_i$ still belongs to this cluster (and below which $\mathbf{x}_i$ is labeled as noise), $\varepsilon(\mathbf{x}_i)$, and the lowest radius at which this cluster or any of its subclusters still exist (and below which all its objects are labeled as noise), $\varepsilon_{\max}(\mathbf{x}_i)$. Notice that this information is directly available during the construction of the density-based clustering hierarchy. This means that, if we keep track and store this information while running HDBSCAN*, then Equation (8) can be computed subsequently in constant time for each single object, which results in an overall complexity of $O(n)$ for computing GLOSH for all n objects. A high-level pseudocode is presented in Algorithm 4.

Note that for objects inside a cluster following, for example, a uniform distribution, their densities tend to be at least approximately the same as the density of the densest objects in the cluster and, therefore, their degree of outlierness according to GLOSH will be all close to zero, thus suggesting that such objects are inliers. If the cluster

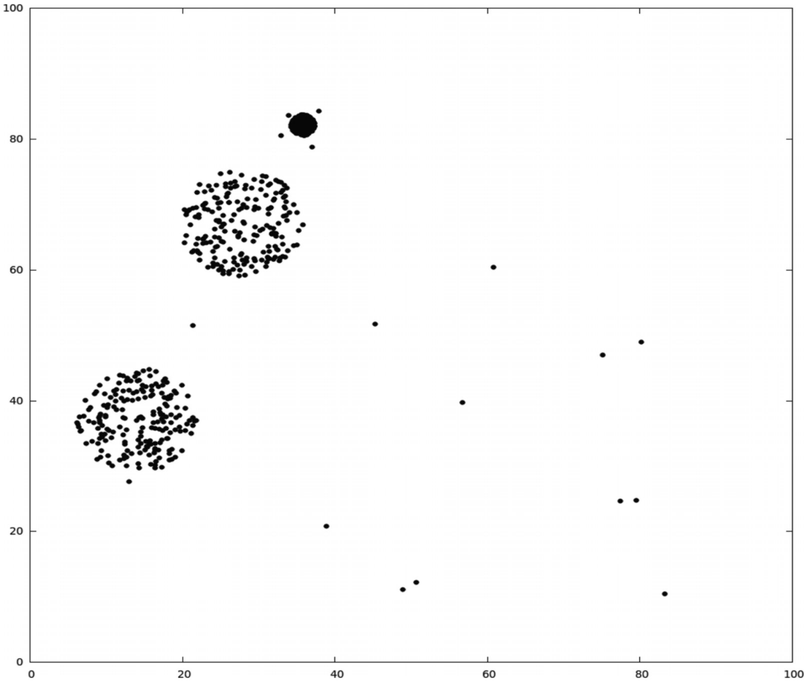


Fig. 10. Example dataset with global and local outliers.

ALGORITHM 4: Main steps for computing GLOSH scores

```
1. While building the HDBSCAN* clustering hierarchy top-down, keep track and store the
following information for each object $\mathbf{x}_i$:
    • The last cluster, say $\mathbf{C}_j$, to which object $\mathbf{x}_i$ belongs top-down through the hierarchy,
    that is, before it becomes noise.
    • The lowest radius $\varepsilon$, denoted herein as $\varepsilon(\mathbf{x}_i)$, at which $\mathbf{x}_i$ still belongs to cluster $\mathbf{C}_j$
    (and below which $\mathbf{x}_i$ is labeled as noise).
    • The lowest radius $\varepsilon$, denoted herein as $\varepsilon_{max}(\mathbf{x}_i)$, at which cluster $\mathbf{C}_j$ or any of its
    subclusters still exist (and below which all its objects are labeled as noise).
2. For each object $\mathbf{x}_i$ do:
    2.1 Compute GLOSH($\mathbf{x}_i$) using Equation (8) with the values $\varepsilon(\mathbf{x}_i)$ and $\varepsilon_{\max}(\mathbf{x}_i)$ from
    Step 1.
```

instead follows, for example, a bell-shaped distribution (such as a Gaussian), then the value of GLOSH is expected to be close to zero for those objects within the region of the peak of the distribution, whereas it tends asymptotically to one for objects farther and farther away from that region. This behaviour of GLOSH captures the intended, statistical intuition using a nonparametric model, that is, without making any specific assumption about the distribution of the data.

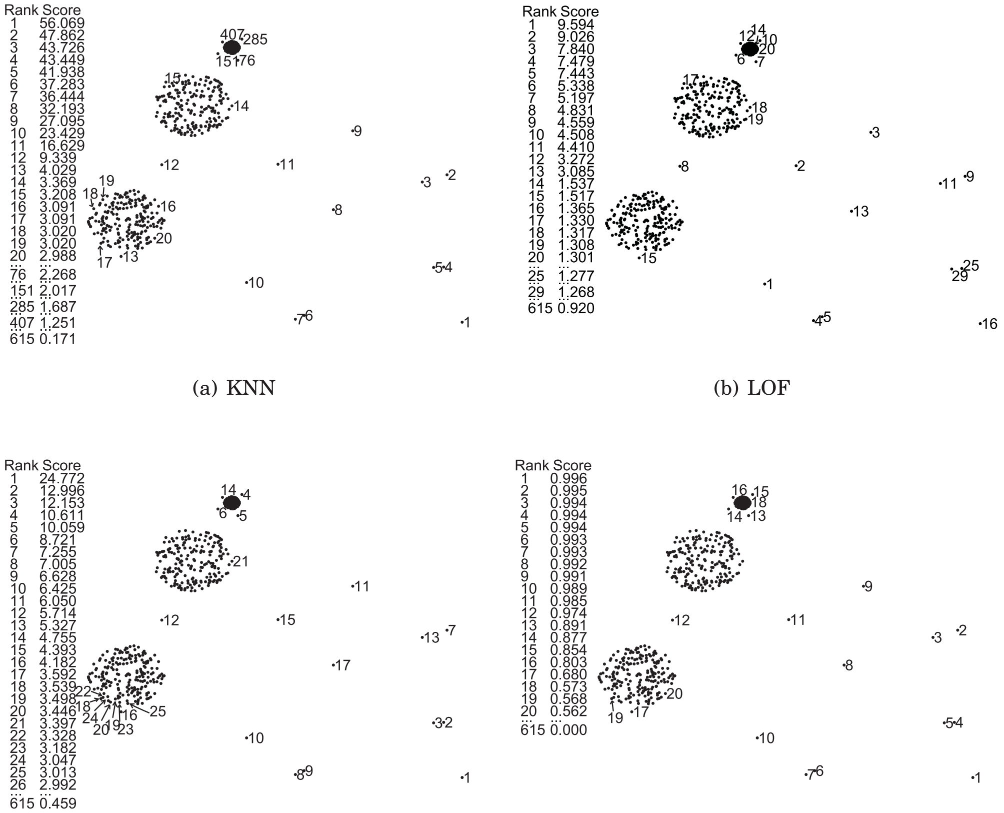
To illustrate the superior behavior of GLOSH, in comparison to representative, state-of-the-art methods, we use the two-dimensional dataset shown in Figure 10. The dataset represents a simple yet paradigmatic example that can illuminate the behaviour of the different approaches to outlier detection. It contains 617 points, distributed over three clusters with different densities of about 200 points each, plus 17 global and local outliers. One can clearly identify four local outliers around the dense cluster at the top of the figure, an outlier between the two less dense clusters, which may be considered local or global, and a number of points in the right part of the figure, which are global outliers, perhaps to different degrees; there is also a point to "the bottom" of the lower left cluster for which a decision, whether it is a local outlier or not, is not that "clear cut" since the cluster is not very dense compared to the distance of the point from the cluster, yet the point seems to be farther outside of the clusters than any other point belonging to the cluster.

It is very unlikely that for this example different people would 100% agree on a particular ranking of all these outliers. However, without looking at the results of any method, we would argue that most people will agree with the following properties of a ranking of these outliers by an "ideal" method: (1) the global outliers should be ranked before the local outliers; (2) the global outliers should be assigned similarly high outlier scores, ideally somewhat higher, the further away they are from the clusters, with the bottom right point having the highest outlier score, and then slightly lower scores, in some order, "moving" from the periphery inwards towards the clusters, with the point at about the center of the data space, closest to the clusters, having the lowest of the global outliers scores; (3) the point between the lower density clusters should then be ranked next with a score that is "in-between" the clearly global and the clearly local outliers; (4) the clearly local outliers around the dense cluster should be ranked next with similar scores, a clear gap to the scores of the global outliers, and ideally ranked slightly higher the further they are away from the dense cluster; (5) the next point in the ranking should then be the point at the bottom of the lowest cluster, with a score that is still distinguishable from the scores of points inside of clusters, but to a much lesser degree than the scores of the local outliers around the dense cluster; (6) after that, the ranking should list the scores of points inside of clusters in some order that starts from least dense points and ends with the densest points, showing scores that are clearly distinguishable from the previous (higher) outlier scores, yet not showing anymore a significant gap between consecutive scores for these points in clusters.

Let us now compare the actual results of some previous, representative outlier detection methods with the results of GLOSH. The methods we show here are: (1) KNN [Ramaswamy et al. 2000] as a representative "global method", (2) LOF [Breunig et al. 2000] as a representative "local method", and (3) LOCI [Papadimitriou et al. 2003] as a method that, with high computational cost, looks at multiple scales to determine outliers. We set the parameters of the different methods as following: $m_{\text{pts}} = 10$ (for LOF and GLOSH) and $k = 10$ for KNN; for LOCI we use the default values, resulting in $r_{\max} = 176.6$, which consequently, for every point, includes the whole dataset in the sampling and counting radius, for the largest considered scale. The results for KNN, LOF, and LOCI are obtained by using the implementations available in ELKI [Achtert et al. 2013]. For each method, we indicate in Figure 11 the rank assigned to the discussed outlier points (plus some additional points in some cases), as well as part of the ranking, showing the actual scores assigned to points.

Figure 11(a) shows the result for KNN. KNN finds all the global outliers, the outlier "in-between" the less dense clusters, and perhaps the point at the bottom of the lowest cluster in a way that is consistent with the earlier formulated requirements. However, it fails entirely to distinguish the local outliers around the dense cluster from other points. Their outlier scores are lower than the scores of many points of the less dense clusters, with ranks as low as 285 and 407. For a "global method", this is the expected behavior and the motivation for the design of "local methods".

Figure 11(b) shows the result for LOF. While LOF finds all the local outliers with clear outlier scores and some of the global outliers at the top of the ranking, the order of the outliers is not at all consistent with the earlier formulated requirements. The top of the ranking consists of a mix of local and global outliers; the most extreme outlier



Rank	Score
1	56.069
2	47.862
3	43.726
4	43.449
5	41.938
6	37.283
7	36.444
8	32.193
9	27.095
10	23.429
11	16.629
12	9.339
13	4.029
14	3.369
15	3.208
16	3.091
17	3.091
18	3.020
19	3.020
20	2.988
...	...
76	2.268
...	...
151	2.017
...	...
285	1.687
...	...
407	1.251
...	...
615	0.171

(a) KNN

Rank	Score
1	9.594
2	9.026
3	7.840
4	7.479
5	7.443
6	5.338
7	5.197
8	4.831
9	4.559
10	4.508
11	4.410
12	3.272
13	3.085
14	1.537
15	1.517
16	1.365
17	1.330
18	1.317
19	1.308
20	1.301
...	...
25	1.277
...	...
29	1.268
...	...
615	0.920

(b) LOF

Rank	Score
1	24.772
2	12.996
3	12.153
4	10.611
5	10.059
6	8.721
7	7.255
8	7.005
9	6.628
10	6.425
11	6.050
12	5.714
13	5.327
14	4.755
15	4.393
16	4.182
17	3.592
18	3.539
19	3.498
20	3.446
21	3.397
22	3.328
23	3.182
24	3.047
25	3.013
26	2.992
...	...
615	0.459

(c) LOCI

Rank	Score
1	0.996
2	0.995
3	0.994
4	0.994
5	0.994
6	0.993
7	0.993
8	0.992
9	0.991
10	0.989
11	0.985
12	0.974
13	0.891
14	0.877
15	0.854
16	0.803
17	0.680
18	0.573
19	0.568
20	0.562
...	...
615	0.000

(d) GLOSH

Fig. 11. Outlier scores produced by the different methods KNN, LOF, LOCI, and GLOSH, for a simple 2-dimensional data set. Numbers beside points stand for outlier ranks.

to the bottom right, is only listed at position 16 with a much lower score than, for example, the point at the approximate center of the data space, which (according to the formulated requirements) should be ranked after the other global outliers, yet is listed by LOF at position 2. LOF also fails completely to distinguish the global outliers at the bottom right (points ranked 25 and 29) from points in clusters—there is no clear gap in outlier scores that would indicate that these two points are stronger outliers than points inside the less dense clusters. This behavior is a typical drawback of "local methods".

Figure 11(c) shows the result for LOCI. Note that LOCI, as originally proposed, does not return a ranking but labels a point p as outlier, if for some scale (radius) r the value $\mathrm{MDEF}(p, r, \alpha)$ for a point p is larger than k times the normalized deviation of the MDEF values of the points in the sampling neighborhood of p, $\sigma_{\mathrm{MDEF}}(p, r, \alpha)$. Papadimitriou et al. [2003] use a fixed cutoff value $k = 3$, but one can naturally return as an outlier score the value $k = {}^{\mathrm{MDEF}(p,r,\alpha)}/_{\sigma_{\mathrm{MDEF}}(p,r,\alpha)}$ for the scale r at which k is maximal. We use the implementation in ELKI [Achtert et al. 2013] that provides a ranking based on these values of k as outlier scores. The original LOCI result can be derived from this ranking by labeling all points with a score greater than three as ouliers. Considering just the ranking, we can observe that LOCI puts all the local and global outliers at the top of the ranking, but, like LOF, the order of the outliers is not at all consistent with the earlier formulated requirements: the top of the ranking is again a mix of local and global outliers. While in this case now, the most extreme outlier to the bottom right is indeed listed at position 1, the global outlier ranked 17 is ranked after the "questionable" local outlier at the bottom of the leftmost cluster (rank 16). What is worse in this case is that the score for point 17 (3.5914) is not really any different than the following scores, which are associated with points inside clusters (and up to rank 25 are all greater than 3).[9]

Figure 11(d) shows the GLOSH result. Different from the other methods, GLOSH satisfies the requirements of a ranking for an ideal method perfectly, in all details. Outliers are ranked in one of the desired orders, consistent with the requirements, and with scores that are clearly distinguishable from scores of points inside clusters (there is no significant gap in scores after position 17, which indicates that those points are inside of clusters). This latter point should not be underestimated as an achievement, when it comes to the application of outlier detection methods to datasets that cannot be depicted in two-dimensional space. In real applications, the actual scores and their ranking is all that is available to determine which points are outliers (and to which degree) and which points are inliers. Accordingly, Hawkins [1980] already claims that *"a sample containing outliers would show up such characteristics as large gaps between 'outlying' and 'inlying' observations and the deviation between outliers and the group of inliers, as measured on some suitably standardized scale"*. Ideal outlier scores should serve as such a *"suitably standardized scale"* but commonly fail to do so, as discussed by Kriegel et al. [2011a]. We argue that the actual scores of GLOSH are easier to interpret and that the differences between the scores make more sense than the differences between the scores of all the other methods. For KNN, LOF, and LOCI alike, there are points with apparently similar "location in space" (and therefore similar expected outlier degree), which have widely different scores, while points in different location, suggesting very different outlier degrees, can have very similar outlier scores assigned to them. This is not the case for GLOSH, which is much more consistent in this respect. Furthermore, when looking at the scores alone, it is not at all clear from the outlier scores in the rankings of KNN, LOF, and LOCI, in which one could draw the line between outliers and inliers and get a result close to the formulated expectation, since there is no clear gap in scores that would separate the outliers from the inliers here.

Note that while this is just a single example to illustrate the properties of different approaches to outlier detection, there is nothing special about this particular dataset and the behavior of the different methods will be similar on similar datasets. A comprehensive experimental evaluation, showing performance summary statistics on a large number of synthetic and real datasets, and comparing GLOSH with a larger number of previous outlier detection methods, is presented in Section 8.

7. RELATED WORK

7.1. Unsupervised Clustering

Density-based clustering is a popular clustering paradigm that is statistically sound and comprises models able to deal with noise, arbitrary shaped clusters, and, possibly, nested subclusters at different density levels [Tan et al. 2006; Ester 2009; Sander 2010]. Despite of the important advances provided by many of the algorithms proposed in this area (e.g., refer to a recent survey of Kriegel et al. [2011]) and in spite of

[9] This case also indicates that LOCI, while trying to bridge the gap between "local" and "global" methods, still behaves in a sense more like a "local" method. For LOF, this point also has a relatively small outlier score among the global outliers, since it is compared to a neighborhood of mostly other outliers.

the widespread use of some of them in practice, the existing density-based clustering techniques have a number of limitations. For instance, most techniques depend on multiple, often critical parameters that must be provided by the user as an input—for example, DBSCAN [Ester et al. 1996], DENCLUE [Hinneburg and Keim 1998, 2003; Hinneburg and Gabriel 2007], DECODE [Pei et al. 2009], and others [Coomans and Massart 1981; Comaniciu and Meer 1999; Ertöz et al. 2002, 2003; Stuetzle 2003; Sander et al. 2003; Brecheisen et al. 2004; Pascual et al. 2006; Stuetzle and Nugent 2010]; many are limited to specific classes of problems, such as complex networks [Sun et al. 2010] or point sets in the real coordinate space [Daszykowski et al. 2001; Foss and Zaïane 2002; Pei et al. 2006, 2009; Stuetzle and Nugent 2010].

An important limitation of several existing density-based clustering algorithms is that they can *only* provide a flat labeling of the data objects, usually based on the use of a single, global density threshold that discriminates between objects lying in dense versus non-dense areas (e.g., Ester et al. [1996], Hinneburg and Keim [1998], Daszykowski et al. [2001], Foss and Zaïane [2002], Pei et al. [2006], Hinneburg and Gabriel [2007], and Cuevas et al. [2000, 2001]). Setting the density threshold is challenging because small changes may significantly affect the clustering results, which makes the selection of an appropriate value particularly difficult. In light of this fact, some strategies (mostly heuristics) have been proposed that are aimed at determining a suitable value for such a threshold [Ester et al. 1996; Daszykowski et al. 2001; Foss and Zaïane 2002; Pei et al. 2006]. Without going into the merits of the performance of these strategies in real applications, it is worth noticing that they do not address another important problem: often, it is not possible to simultaneously detect clusters of varied densities by using a single density threshold that attempts to separate these clusters from noise, as such a single threshold may not exist—for example, see the work of Ankerst et al. [1999], Stuetzle [2003], Tan et al. [2006], and Kriegel et al. [2011] for discussions and illustrative examples. Methods such as those described by Ertöz et al. [2002], Liu et al. [2007], and Pei et al. [2009] are claimed to be able to detect clusters with different densities in data. However, they cannot describe possible hierarchical relationships that may exist between nested clusters lying on different density levels, which is only possible by using hierarchical density-based clustering techniques [Wishart 1969; Ankerst et al. 1999; Sander et al. 2003; Stuetzle 2003; Brecheisen et al. 2004; Stuetzle and Nugent [2010; Sun et al. 2010; Gupta et al. 2010].

Apart from methods aimed at getting approximate estimates of density-contour trees for continuous-valued PDF's—for example, see the work of Stuetzle and Nugent [2010] and references therein—not much attention has been given to hierarchical density-based clustering in more general data spaces. Ankerst et al. [1999] proposed the OPTICS algorithm whose primary output is the so-called *reachability plot*, discussed in Section 4. Besides being a useful visualization tool for cluster analysis on its own, the main conceptual appeal behind this sort of graphical representation is that, for a given value of m_{pts}, a single plot encodes all possible DBSCAN-like clustering solutions w.r.t. the parameter ε (except for eventual differences in the assignment of border objects), in a nested way. Reachability plots are, therefore, a particular kind of hierarchical clustering representations. A horizontal cut through the plot corresponds to a DBSCAN-like clustering for a specific value of ε, the only possible differences referring to the assignment of border objects. In the same article, Ankerst et al. [1999] also proposed a postprocessing procedure to extract a simplified cluster tree from the reachability plot produced by the OPTICS algorithm. This procedure did not become as popular as OPTICS itself, probably because it is very sensitive to the choice of a critical parameter that cannot easily be determined or understood. Moreover, no automatic method to extract a flat clustering solution based on local cuts in the obtained tree was described.

Sander et al. [2003] proposed an improved method to extract trees of significant clusters from reachability plots. This method is less sensitive to the user settings than the original method of Ankerst et al. [1999]. However, this method is based on heuristics with embedded threshold values that can still affect the results significantly, and the problem of extracting a flat solution from local cuts in the cluster tree was practically untouched; the only mentioned (ad-hoc) approach was to arbitrarily take all the leaf clusters and discard the others. This is also the approach adopted by Stuetzle [2003], which in that case is equivalent to performing local cuts through a conventional single-linkage dendrogram (though the formalism behind that method is based on density estimates). Sun et al. [2010] recompiled the original findings from Ankerst et al. [1999], Sander et al. [2003], and Lelis and Sander [2009] in the particular context of community discovery in complex networks. However, no mechanism to extract a simplified cluster tree from the resulting clustering dendrogram was adopted, and only a method producing a global cut through the dendrogram was described. Brecheisen et al. [2004] proposed another method to extract trees of significant clusters from reachability plots that was claimed to be better than the method of Sander et al. [2003]. The work is focused on interactive visualization as a means to explore the obtained hierarchy; the problem of extracting flat solutions is not discussed.

The algorithm AUTO-HDS [Gupta et al. 2010] is, like our method, based on a principle used to simplify clustering hierarchies, which in part refers back to the work of Herbin et al. [2001]. The clustering hierarchy obtained by AUTO-HDS is typically a subset of the one obtained by our method HDBSCAN*. Conceptually, it is equivalent to a sampling of the HDBSCAN* hierarchical levels, from top to bottom, at a geometric rate controlled by a user-defined parameter, r_{shave}. Such a sampling can lead to an underestimation of the stability of clusters or even to missed clusters, and these side effects can only be prevented if $r_{\text{shave}} \to 0$. In this case, however, the asymptotic running time of AUTO-HDS is $O(n^3)$ [Gupta et al. 2006] (in contrast to $O(n^2 \log n)$ for "sufficiently large" values of r_{shave})[10], which is the same as the complexity required by Wishart's HMA method [Wishart 1969] to produce an equivalent hierarchy. In addition, the stability measure used in AUTO-HDS has the undesirable property that the stability value for a cluster in one branch of the hierarchy can be affected by the density and cardinality of other clusters lying on different branches. AUTO-HDS also attempts to perform local cuts through the hierarchy in order to extract a flat clustering solution, but it uses a greedy heuristic for selecting clusters that may give suboptimal results in terms of an overall stability.

The HDBSCAN* algorithm, including the postprocessing procedure to optimally extract a flat clustering solution in the *unsupervised* scenario, was preliminarily introduced by Campello et al. [2013a]. This previous article lacks, however, detailed explanations, examples, complexity analyses, a comprehensive related work, and all the other extensions and discussions presented here (visualization, semisupervision, and outlier detection).

7.2. Semisupervised Clustering

In the semisupervised domain, the works of Ruiz et al. [2007], Böhm and Plant [2008], Hang et al. [2009], Lelis and Sander [2009], and Ruiz et al. [2010] are related to ours in the sense that they also perform semisupervised density-based clustering. Ruiz et al. [2007, 2010] proposed a modified version of DBSCAN, named C-DBSCAN, that can handle instance-level constraints. The main shortcomings of this method are twofold. First, like DBSCAN, C-DBSCAN uses a single, global density threshold, which

[10] Further gains have been shown to be possible by using parallel computing techniques, but this is possible for AUTO-HDS only in very low dimensional spaces [Dhandapani et al. 2010].

is determined by a particular pair of values of the parameters m_{pts} and ε. Second, the algorithm deals with the constraints in a hard sense, that is, clusters under cannot-link constraints are not allowed to be formed and different clusters under must-link constraints are forced to be merged. As a consequence, in order to completely satisfy the constraints provided by the user, the algorithm indiscriminately violates the primary assumption behind the clustering model adopted, namely, the definition of density-based clusters. Hang et al. [2009] proposed a hybrid (prototype- and density-based) heuristic algorithm partially inspired by the DBSCAN and OPTICS idea of cluster expansion as an alternative to incorporate should-link constraints in the process of clustering biological datasets. This method requires three user-defined parameters to be set, and it cannot handle should-not-link constraints. Böhm and Plant [2008] presented HISSCLU, a hierarchical density-based clustering algorithm based on OPTICS. Given a set of labeled objects, HISSCLU starts the OPTICS expansion simultaneously from all the labeled objects and generates as many reachability plots as the number of labeled objects, each one representing a cluster. During the label expansion they use a method to change the distance between objects that resembles distance learning [Bilenko et al. 2004]. The reachability plots are reordered and concatenated with each other, producing one single plot. The result of HISSCLU is a cluster hierarchy. The authors do not propose a particular approach to extract clusters from the hierarchy. In their experiments, in which a flat clustering was required for comparisons with other methods, they performed a horizontal cut at a certain level through the plot to extract the clusters. In this setting, the method has the same limitations as DBSCAN in regard to the use of a single, global density threshold. In order to circumvent this limitation, the algorithm SS-DBSCAN was proposed by Lelis and Sander [2009]. Similarly to HISSCLU, SS-DBSCAN also starts the OPTICS expansion simultaneously from all the labeled objects, but it stops an expansion as soon as a violation of a cannot-link constraint induced by the labels is detected. The resulting clusters may correspond to local cuts through the OPTICS hierarchy at different density levels, but such levels are determined solely in terms of the induced cannot-link constraints, neither must-link constraints nor unsupervised measures of cluster quality (e.g., stability) are considered. Moreover, an underlying assumption of the method is that there is at least one labeled object of each data category to be discovered by the algorithm as a cluster. As as consequence, the method cannot discover natural clusters whose objects are not involved in the partial information provided *a priori* by the user.

A general framework and some instances of this framework for extracting flat clustering solutions from cluster trees of varied natures, possibly in a semisupervised way, was introduced by Campello et al. [2013b]. The semisupervised version of the method for the particular instance of HDBSCAN* hierarchies, however, was not discussed or experimentally evaluated.

7.3. Outlier Detection

The work of Knorr and Ng on the distance-based notion of outliers (DB-outlier) [Knorr and Ng 1997b, 1998; Knorr et al. 2000] tried to reconcile statistical parametric approaches with efficient database-oriented approaches and triggered the data mining community to develop many different approaches that have a less statistically oriented but more spatially oriented notion to model outliers. The DB-outlier model relies on the choice of two thresholds, D and p. In a set of objects (for these methods, usually a database of real-valued feature vectors) O, an object $o \in O$ is an outlier if at least a fraction p of all data objects in O are not contained within a radius D from o. This idea is based on statistical reasoning [Knorr and Ng 1997a] but simplifies the approach to outlier detection considerably, motivated by the need for scalable methods handling large datasets.

The K-NN-outlier model [Ramaswamy et al. 2000] uses the distances to the K-NNs and ranks the objects according to their distances to their K-th NN. A partition-based algorithm is then used to efficiently mine top-n outliers. As a variant, the K-NN-weight model [Angiulli and Pizzuti 2002] uses the sum of distances to all objects within the set of K-nearest NN (called the weight) as an outlier degree along with an approximation solution, based on space filling curves, to enable scalability with increasing size of the dataset.

Work following these approaches was primarily interested in algorithmically improving efficiency, for example based on approximations or improved pruning techniques for mining the top-n outliers [Bay and Schwabacher 2003; Kollios et al. 2003; Nguyen and Gopalkrishnan 2009; Angiulli and Fassetti 2009]. Several efficient or approximate algorithms for mining distance-based outliers have been studied by Orair et al. [2010]. They identify common algorithmic techniques but do not discuss model properties, restricting themselves to the distance-based models [Knorr and Ng 1997b; Ramaswamy et al. 2000; Angiulli and Pizzuti 2002].

While in these distance-based approaches for each object a property (outlier model) [Schubert et al. 2014b] is learned based on a local neighborhood (radius ε, KNNs), the objects are eventually ranked according to this property ("outlier score") in a global way. For example, the object with the largest K-NN distance overall would be the most prominent outlier. Thus, these methods are best suited to identify *global* outliers. Recent global approaches base the decision not on Euclidean distances but on angle-variance (ABOD [Kriegel et al. 2008] and an efficient variant using random projections [Pham and Pagh 2012]).

Identifying *local* outliers (i.e., comparing local models with a local reference set [Schubert et al. 2014b]) started with the method LOF [Breunig et al. 2000]. The basic idea is to assign a local density estimate (local reachability density, *lrd*) to each object of the database. Then, LOF considers ratios between the *lrd* of an object and the *lrd*s of its neighboring objects. Thus, the resulting outlier score is based on a local comparison rather than on a global comparison. A LOF value of approximately 1 indicates that the corresponding object is located within a region of homogeneous density (i.e., a cluster). If the difference between the density in the local neighborhood of o and the density around the K-NNs of o is higher, o gets assigned a higher LOF value. The higher the LOF value of an object o is, the more distinctly is o considered an outlier.

Several extensions and refinements of the basic LOF model have been proposed, for example a COF (connectivity-based outlier factor) [Tang et al. 2002], or using the concept of micro-clusters to efficiently mine the top-n density-based local outliers in large databases (i.e., those n objects having the highest LOF value) [Jin et al. 2001]. A similar algorithm, named INFLO [Jin et al. 2006], for an extension of the LOF model is using also the reverse NNs additionally to the NNs and considering a symmetric relationship between both values as a measure of outlierness. The local distance-based outlier detection (LDOF) approach [Zhang et al. 2009] merges the notion of local outlierness with the distance-based notion of outliers. LoOP [Kriegel et al. 2009a] uses a density estimation based on the distance distribution of all NNs and formulates the local outlier score as a probability. COP [Kriegel et al. 2012] aims at detecting outliers in the presence of local correlations in the dataset by measuring the deviation from the local model. A variant of LOF for high-dimensional data de [Vries et al. 2010, 2012] uses random projections to find anomalies approximately with a certain probability. KDEOS [Schubert et al. 2014a] generalizes local density-based approaches, discussing the proper way of incorporating kernel density estimates. Some specialized approaches for high dimensional data try to account for a local feature relevance and search outliers in subspaces of the data space [Aggarwal and Yu 2001; Kriegel et al. 2009b; Müller et al. 2008, 2010, 2011, 2012; Nguyen et al. 2011; Keller et al. 2012; Micenková et al.

2013; Dang et al. 2014], see the survey of Zimek et al. [2012]. More or less explicitly, all these methods basically aim at providing rather simple approximations of statistical density estimates around objects.

Let us note that some methods have been proposed to combine single outlier detectors into an ensemble [Lazarevic and Kumar 2005; Gao and Tan 2006; Nguyen et al. 2010; Kriegel et al. 2011a; Schubert et al. 2012; Liu et al. 2012; Zimek et al. 2013a, 2013, 2014]. Under certain conditions [Kriegel et al. 2011a; Schubert et al. 2012] it becomes meaningful to combine even different methods and therefore to run global methods and local methods in combination. But such a combination, by selecting methods and weighting schemata, would necessarily also put emphasis either on local or on global outliers.

None of the single approaches so far can find simultaneously global and local outliers. The only exception is an outlier detection schema, named Local Correlation Integral (LOCI) [Papadimitriou et al. 2003], which, at a very high computational cost, is to some degree successful in finding both local and global outliers. LOCI is based on the concept of a multi-granularity deviation factor (MDEF), and compares the point density of a point to densities of neighboring points at different distances. A point is selected as an outlier if its MDEF value at some radius r deviates significantly from the MDEF values of its neighbors at radius r. The advantages of LOCI are that the most expressive neighborhood for each point is determined dynamically, and may lead to the detection of both local and global outliers. Global outliers can be detected if MDEF values are considered for all radii up to a radius which includes the whole dataset for each point, but with this parameter setting, the computational complexity of LOCI is prohibitive for large datasets; it is $O(n^3)$ when using, as suggested by the authors, a maximum sampling radius $r_{\max}$ that essentially includes the whole dataset, where n is the number of objects. A faster, approximate method, aLOCI, which was also introduced by Papadimitriou et al. [2003], solves this problem only partially since, in practice, the approximation seems to be often too loose (as we show in one of our experiments in Section 8.2).

Some approaches to outlier detection also, in some sense, base the choice of a reference set for potential outliers on a preceding clustering step. Objects that are not covered by the clusters (or by a sufficient number of clusters in case of applying multiple clustering procedures or a clustering ensemble) are then deemed outliers [Müller et al. 2008, 2012]. These solutions depend on arbitrarily chosen density thresholds (or other parameters for the clustering process) and unduly many objects may become outliers without a convincing way to deduct a meaningful ranking [Zimek et al. 2012]. As a result, the complete, flat clustering is taken into account, and the perspective on outlierness is global rather than adaptive to local characteristics.

8. EXPERIMENTS AND DISCUSSION

8.1. Clustering

Datasets. We used real datasets with a variety of characteristics (no. of objects, dimensionality, and no. of clusters) and from different domains, namely, text, biology, image, and UCI datasets. Two datasets, “Articles-1442-5” and “Articles-1442-80”, consist of high dimensional representations of text documents. They are made available on request by Naldi et al. [2011] and are formed by 253 articles represented by 4,636 and 388 dimensions, respectively, both with five classes. We used the Cosine measure as dissimilarity function for these datasets. Two datasets, “CellCycle-237” (made public by Yeung et al. [2001]) and “YeastGalactose” (used by Yeung et al. [2003]) represent gene-expression data and contain 237 respectively 205 objects (genes), 17 respectively 20 dimensions (conditions), and 4 known classes. For these datasets we used

Euclidean distance on the z-score normalized objects, which is equivalent to using Pearson correlation on the original data. Two datasets, “Iris” and “Ecoli”, are from the UCI Repository [Bache and Lichman 2013]. They contain 150 and 336 objects in 4 and 7 dimensions, with 3 and 8 classes, respectively. For these datasets we used Euclidean distance.

In addition to individual datasets, we also report average performance on two dataset collections based on the Amsterdam Library of Object Images (ALOI) [Geusebroek et al. 2005]. Image sets were created as in [Horta and Campello 2012] by randomly selecting k ALOI image categories as class labels 100 times for each $k = 2, 3, 4, 5$, then sampling (without replacement), each time, 25 images from each of the k selected categories, thus resulting in 400 sets, each of which contains 2, 3, 4, or 5 clusters and 50, 75, 100, or 125 images (objects). The images were represented using six different descriptors: color moments, texture statistics from the gray-level co-occurrence matrix, Sobel edge histogram, first order statistics from the gray-level histogram, gray-level run-length matrix features, and gray-level histogram, with 144, 88, 128, 5, 44, and 256 attributes, respectively. We report results for the texture statistics (as a typical case), denoted by “ALOI-TS88”, and for a six-dimensional representation combining the first principal component extracted from each of the six descriptors using PCA, denoted by “ALOI-PCA”. We used Euclidean distance in both cases.

Evaluation Measures. We report the well-known Adjusted Rand Index (ARI) [Hubert and Arabie 1985] in all experiments. ARI essentially measures the fraction of pairs of objects that are in agreement (either clustered or not clustered together) both in the ground truth partition and in a given partition to be evaluated, then it adjusts this measurement for chance by removing its expected value under the hypothesis of random partitions [Jain and Dubes 1988; Everitt et al. 2001]. We have also computed the Overall F-measure [Larsen and Aone 1999], but the results are omitted for the sake of compactness as the conclusions that can be drawn are the same as for ARI. Both ARI and F-measure are measures commonly employed in the literature and they have been computed here *not* taking into account those data objects involved in constraints when constraints are used by semisupervised algorithms under evaluation.

Algorithms. Our method, HDBSCAN*, is compared with the following algorithms: (i) AUTO-HDS [Gupta et al. 2010], which is also based on local cuts through a density-based clustering hierarchy; (ii) the method in Sander et al. [2003], referred to here as “OPTICS-AutoCl”, which consists of the selection of the leaf nodes of a density-based cluster tree extracted from an OPTICS reachability plot; and (iii) SS-DBSCAN [Lelis and Sander 2009], which is a constrained clustering procedure also based on OPTICS. The first two algorithms are unsupervised, while the third one is semisupervised.

Experimental Settings. For OPTICS-AutoCl and SS-DBSCAN, the optional OPTICS speedup control value ε was not used so that we get exact rather than truncated results ($\varepsilon =$ “infinity”). These algorithms demand a smoothing parameter, $MinPts$, which plays precisely the same role as m_{pts} in HDBSCAN*. AUTO-HDS has also an equivalent parameter, n_{ε}. We set such a noncritical parameter to $MinPts = m_{pts} = n_{\varepsilon} = 4$ in all algorithms and experiments; this value has also been used in experiments with AUTO-HDS in Gupta et al. [2010]. The other parameters required by OPTICS-AutoCl were set as suggested by the authors in the original reference [Sander et al. 2003]. For AUTO-HDS, we also set the shaving rate r_{shave} to 0.03, following the authors’ suggestion to use values between 0.01 and 0.05. For HDBSCAN*, we use the hierarchy compaction procedure with minimum cluster size $m_{clSize} = m_{pts}$, as discussed in Section 3.3. AUTO-HDS has a related parameter to control the maximum size of clusters deemed spurious, n_{part}, which we set equivalently as $n_{\text{part}} = n_{\varepsilon} - 1$. We also ran experiments for other values of m_{pts} ($MinPts$, n_{ε}), m_{clSize} (n_{part}), and r_{shave}, and the main conclusions do not change.

We have performed experiments in both unsupervised and semisupervised scenarios. The partial information for the experiments in the semisupervised scenario was obtained in the form of labeled objects randomly selected from the datasets. These objects were *not* considered when assessing the quality of the results. SS-DBSCAN uses the labels explicitly. The should-link and should-not-link constraints that can be derived from the labels (when available) are used by HDBSCAN*. We set the number of labeled objects to 0, 2, 4, 6, 8, and 10 percent of the number of objects in each dataset, where 0 corresponds to the unsupervised case. When labeled objects are used, the results reported are averages over 100 random selections of label sets. Note that OPTICS-AutoCl and AUTO-HDS are unsupervised methods, therefore their results do not change with the number of labeled objects available. Contrarily, SS-DBSCAN works only in a semisupervised way, so no results for this algorithm are shown when there are no labeled objects available.

Clustering Results. The results are shown in Figure 12. Notice that the overall results of HDBSCAN* in the unsupervised scenario are quite competitive when compared to OPTICS-AutoCl and AUTO-HDS. In some cases, such as YeastGalactose, Articles -1442-5, and Articles-1442-80, the unsupervised performance leaves not much room for further improvements, and the addition of constraints does not have an effect. In other cases, namely, CellCycle-237, ALOI-TS88, ALOI-PCA, and, in particular, Ecoli, the performance of HDBSCAN* improves by adding constraints.

When compared with SS-DBSCAN, HDBSCAN* provides better results in almost all cases (the only exception is Iris with 4% or more labels); in many cases the results of HDBSCAN* are better by a large margin, and more prominently so with smaller amounts of labeled objects. For some datasets, such as CellCycle-237 and YeastGalactose, HDBSCAN* with no semisupervision outperforms SS-DBSCAN even when the latter uses 10% of labeled objects.

Notice that Figure 12 shows for some datasets a drop in performance of the semisupervised algorithms when larger amounts of constraints are added (e.g., SS-DBSCAN for CellCycle-237 and Ecoli). As a matter of fact, it has been observed and discussed in the semisupervised clustering literature that adding constraints may possibly decrease the performance of clustering algorithms [Davidson et al. 2006]. In our experiments, considering that we have excluded the objects involved in constraints when computing the evaluation measures, this is particularly plausible. HDBSCAN* has shown quite robust results with this respect; essentially, such a behavior is noticeable only for the Iris dataset.

Finally, in order to illustrate how visualization techniques can help interpret clustering hierarchies for high-dimensional data, we show in Figures 13 and 14 the reachability plots derived from the HDBSCAN* hierarchies for some of the datasets that we have experimented earlier in this section. In all these plots the hierarchies correspond to the fully unsupervised scenario, that is, they are those derived without using any labels or constraints.

Figure 13 shows the plots for two datasets from the ALOI-TS88 collection. These datasets are particularly interesting for illustration purposes as their data objects represent pictures; this allows us to show the actual pictures that correspond to objects in different clusters. Starting with Figure 13(a) on top, the reachability plot suggests that there are four clear clusters characterized by the most prominent “dents”, two of them possibly composed of less prominent subclusters. When we look at the pictures that correspond to the objects within each of the four dents, we can see that they do correspond to the four categories that constitute this particular dataset, namely, white

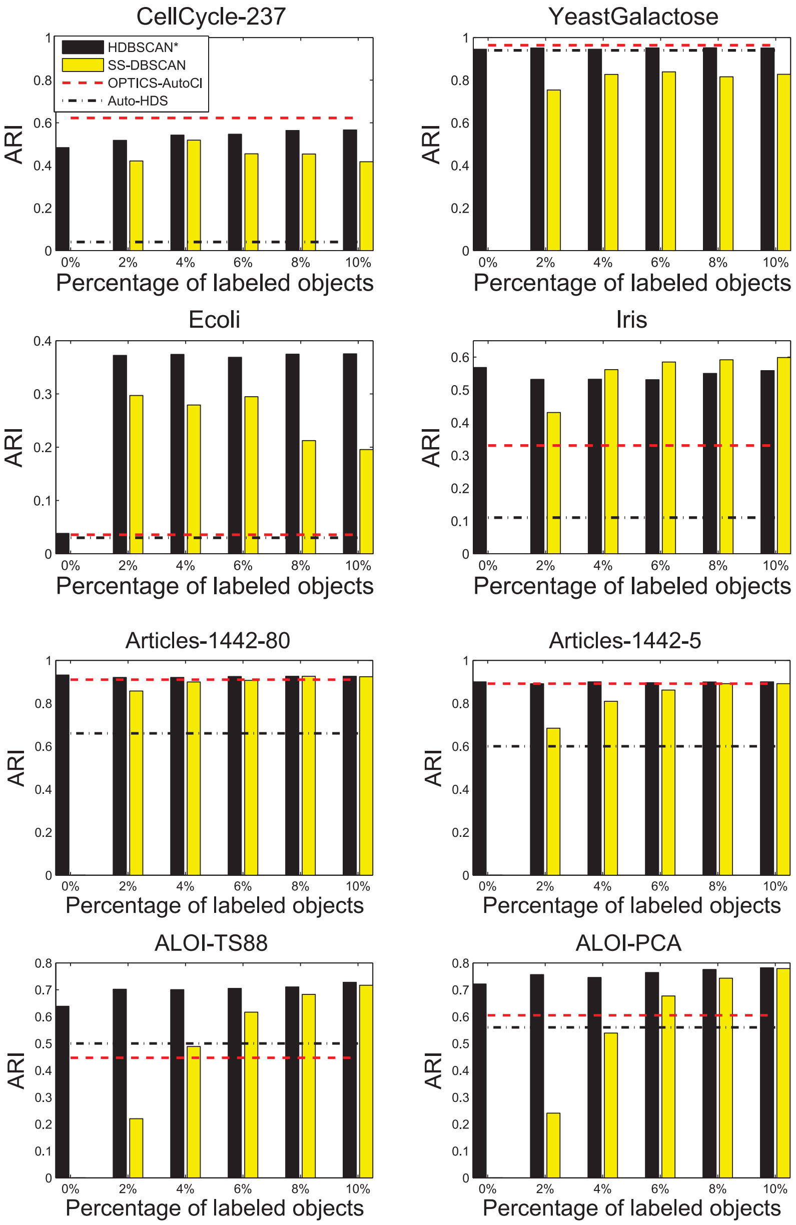
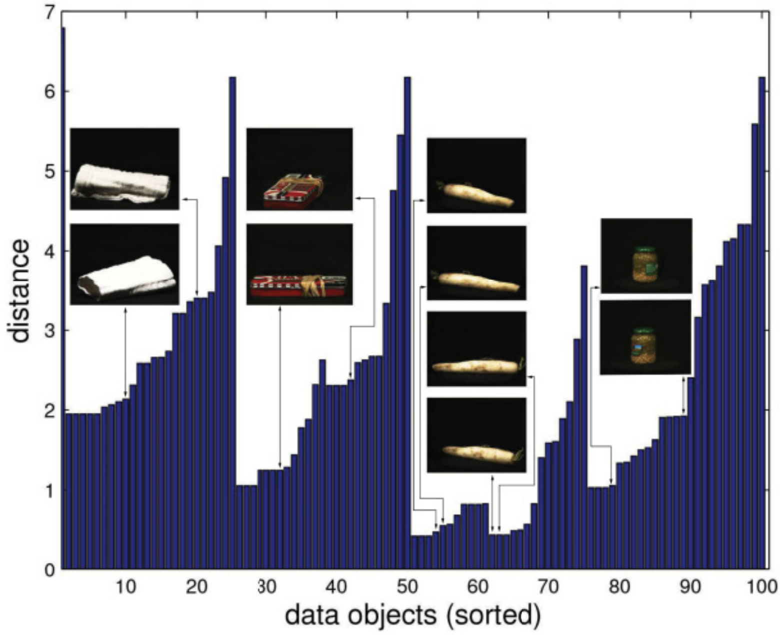
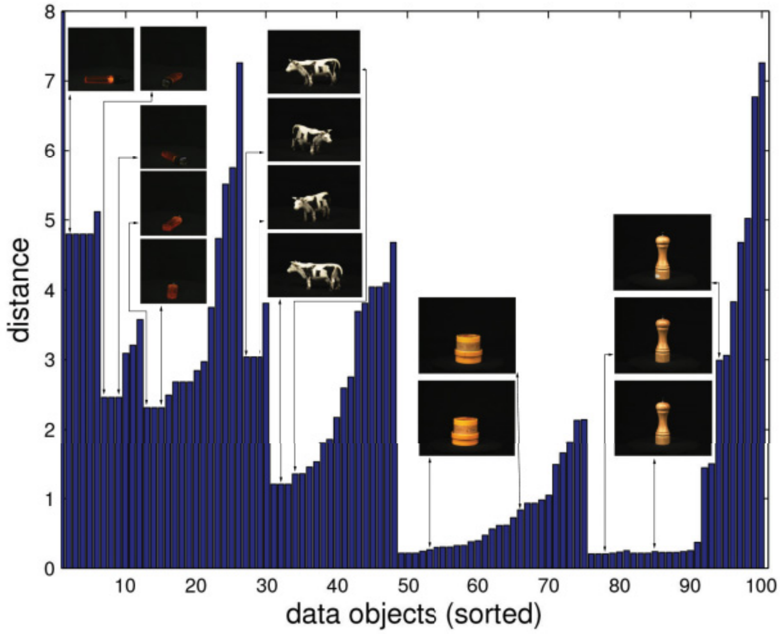


Fig. 12. ARI results for HDBSCAN*, SS-DBSCAN, OPTICS-AutoCl, and AUTO-HDS. SS-DBSCAN is not shown when there are no labeled objects as it does not work in an unsupervised way. OPTICS-AutoCl and AUTO-HDS are shown as lines to emphasize that they are not able to make use of labels, when available.

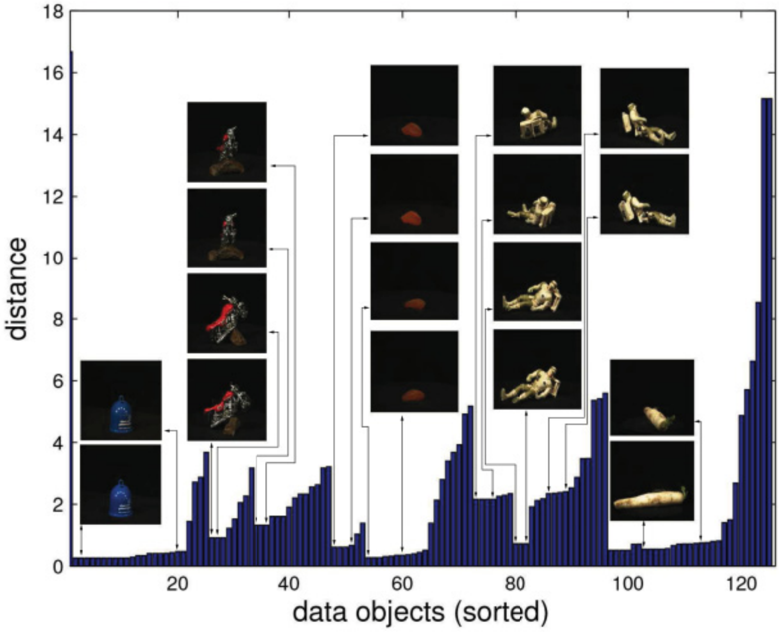


(a)

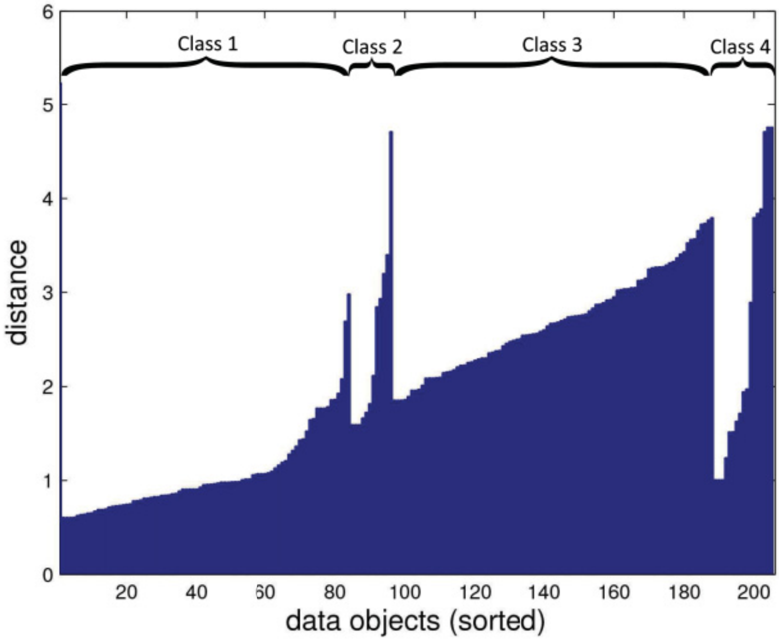


(b)

Fig. 13. Reachability plots of the HDBSCAN* hierarchies for two data sets from the ALOI-TS88 collection.



(a)



(b)

Fig. 14. Reachability plots of the HDBSCAN* hierarchies for two data sets: (a) a data set from the ALOI-TS88 collection; and (b) the YeastGalactose data.

cloths, bike repairs, sweet potatos, and vegetable jars. We can also see that subclusters are characterized mainly by different placements of the bike repairs and sweet potatos (noticeably the side of the stem for the latter). For Figure 13(b) at the bottom the scenario is quite similar: there are also four more prominent dents indicating clusters that do correspond to the four categories of pictures in the respective dataset, namely, lighters, cows, orange boxes and pepper mills, and once again we can see less salient subclusters, in this case for the cows with different angles and for the lighters with different combinations of angles and illumination.

In Figure 14(a) (top) we illustrate a third instance of ALOI-TS88, this one containing five categories that clearly show up as dents in the plot. One of them is subdivided into two noticeable subclusters corresponding to motorcycles that become quite distinctive from different viewing perspectives. At last, in Figure 14(b) (bottom) we show the reachability plot for the YeastGalactose data. Note that the objects belonging to each of the four classes according to the ground truth (known class labels, which have *not* been used during unsupervised clustering) are perfectly captured as four prominent dents in the plot. Note also that classes 3 and 4 are apparently closer to each other and might be seen as subclusters of a bigger cluster. The same holds true for classes 1 and 2. When no information about class labels is available at all, this type of information can be undoubtedly valuable, particularly at the initial stages of exploratory data analysis.

It is worth remarking that the reachability plots derived from the HDBSCAN* hierarchy and displayed in Figures 13 and 14 are equivalent to those that would be obtained by running OPTICS. However, the OPTICS algorithm provides reachability plots only; as discussed, for example, in the related work section, automatically extracting clusters from such plots is a complex task. HDBSCAN*, on the other hand, automatically provides cluster hierarchies from which different plots, including reachability plots, can be trivially extracted, that is, these plots are essentially provided as a byproduct of the algorithm.

8.2. Outlier Detection

Datasets. To test the ability of GLOSH to identify local and global outliers at the same time, we design experiments using synthetic data with different characteristics: We generated 60 synthetic datasets choosing randomly values for the following parameters in the given range: dimensionality $d \in [20, \ldots, 40]$, number of clusters $c \in [2, \ldots, 10]$, for each cluster independently the number of points $n_{c_i} \in [600, \ldots, 1000]$. For each cluster, the points are generated following a Gaussian model as follows: For each cluster c_i, and each attribute a_j, we choose a mean μ_{c_i,a_j} from a uniform distribution in $[-10, 10]$ and a standard deviation σ_{c_i,a_j} from a uniform distribution in $[0.1, 1]$. Then for cluster c_i, n_{c_i} cluster objects (points) are generated attribute-wise by the Gaussians $\mathcal{N}(\mu_{c_i,a_j}, \sigma_{c_i,a_j})$. The resulting cluster is rotated by a series of random rotations and the covariance matrix Σ corresponding to the theoretical model is computed by the corresponding matrix operations [Soler and Chin 1985]. Then, we compute for each point the Mahalanobis distance to its corresponding cluster center, using the covariance matrix Σ of the cluster. For a dataset dimensionality d, the Mahalanobis distances for each cluster follow a χ^2 distribution with d degrees of freedom. We label as outliers those points that exhibit a distance to their cluster center larger than the theoretical 0.975 quantile, independently of the actually occurring Mahalanobis distances of the sampled points. This results in an expected amount of 2.5% *local* outliers per dataset. These datasets have also been used for other recent studies on outlier detection [Zimek et al. 2013a, 2014].

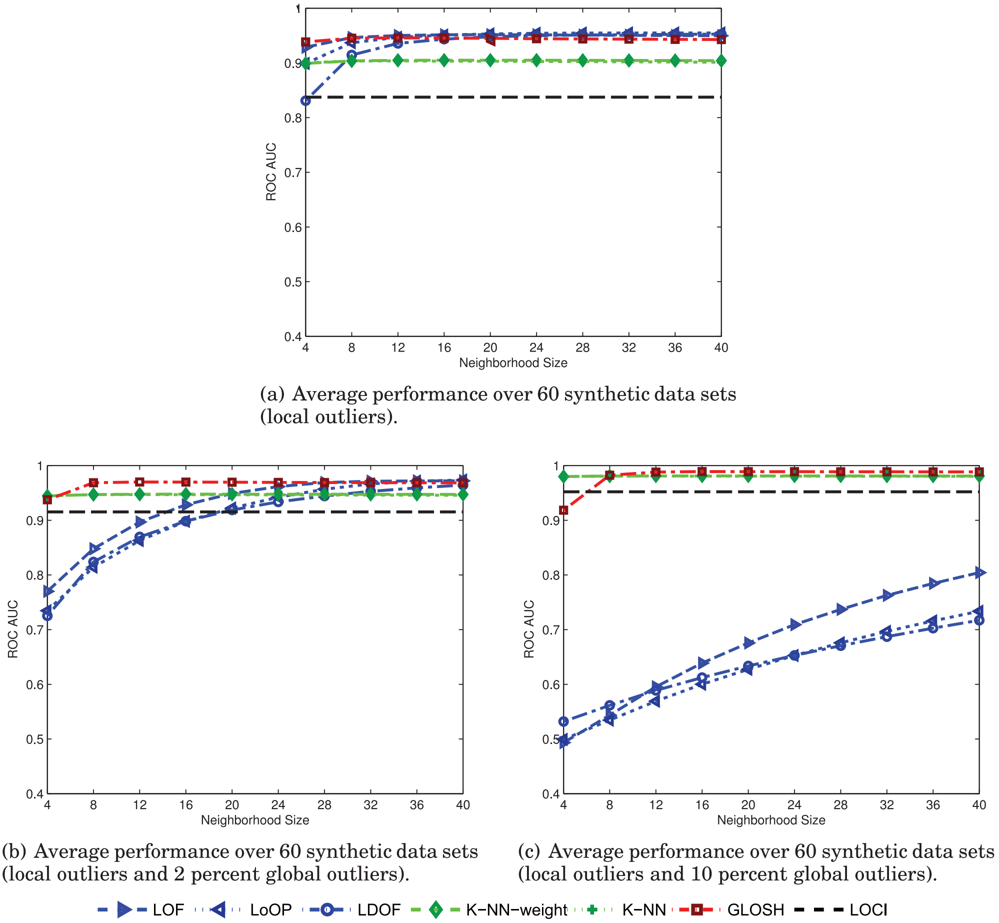
We add global outliers to these datasets, sampling attribute values from a uniform distribution [min, max], where min and max are the minimum and maximum attribute values of the corresponding attribute in the respective dataset. We chose two levels of global noise, resulting in 2% global outliers and 10% global outliers, respectively. Overall, this results in 180 synthetic datasets, in three groups of 60 datasets each (containing 0, 2, and 10 percent of global outliers, respectively).

Additionally, we use real datasets from the UCI repository [Bache and Lichman 2013], with adaptations for the evaluation of outlier detection methods as it is common practice in the literature [Abe et al. 2006; Yang et al. 2008; Kriegel et al. 2008; Zhang et al. 2009; Keller et al. 2012; Zimek et al. 2013a]: In the Lymphography dataset, the very small classes 1 and 4 are combined and treated as outliers against the rest of the classes. In the Pendigits dataset, we use one class (digit) as outlier, sampling 10% of the objects from the outlier class and treating the other, complete, classes as inliers. The same procedure has been applied to the Waveform data. All classes used in turn as outliers show similar results (in waveform almost identical). We show results for class 2 in Pendigits and in Waveform. For Satellite Image, we combined train and test set and transformed the dataset to an outlier task by taking a sample of 10% from class 2. For Segment, we chose classes GRASS, PATH, and SKY, in turn, for downsampling to 10%, which renders the remaining objects of these classes outliers (resulting in three different datasets). Again, the results are similar for these classes and we show PATH only. The KDD Cup 1999 data has been used by Lazarevic and Kumar [2005] in a preprocessed form and since then in many studies on outlier detection in the same way, using “U2R” as outlier class and the “normal” class as inliers. We follow their approach.

Evaluation Measures. A measure of success that is commonly used in evaluation of outlier detection methods [Zimek et al. 2012] is based on receiver operating characteristic (ROC) curves. ROC curves are a plot of the true positive rate against the false positive rate. These plots are usually turned into a measure by computing the area under this curve (AUC), which also allows to display several results in a single graph and to compare the results numerically.

For a random ranking result, both rates (true positive rate and false positive rate) will grow simultaneously, resulting in an area that approximately fills half of the space. For a perfect result that returns all outliers first and only then returns the inliers (i.e., we have 100% true positives before we even get the first false positive), the area under the corresponding curve will cover the available space completely, that is, the maximal ROC AUC value is 1.0. Intuitively, the ROC AUC value can be seen as the probability that a pair of two randomly chosen objects, one positive example (outlier) and one negative example (inlier), is sorted correctly (i.e., the outlier is ranked before the inlier) [Hanley and McNeil 1982]. ROC curves and ROC AUC analysis inherently treat the class imbalance problem by using the relative frequencies which makes them popular for evaluation of outlier detection.

Algorithms. We compare GLOSH to standard local and global methods. As local methods, we chose LOF [Breunig et al. 2000], LDOF [Zhang et al. 2009], and LoOP [Kriegel et al. 2009a]. As global methods, we chose the K-NN outlier model [Ramaswamy et al. 2000] and the K-NN-weight outlier model [Angiulli and Pizzuti 2002]. We also compare to LOCI [Papadimitriou et al. 2003] as the only competitor that also tries to find both types of outliers, global and local ones. On the large KDD Cup 1999 data, we use the approximate version of LOCI, aLOCI [Papadimitriou et al. 2003], as running LOCI on this dataset is infeasible. To allow a comparison in terms of the ranking evaluation, we use the ranking version of LOCI and aLOCI. Note that this is in favor of LOCI and aLOCI since, this way, we avoid counting wrong decisions due



(a) Average performance over 60 synthetic data sets (local outliers).

(b) Average performance over 60 synthetic data sets (local outliers and 2 percent global outliers).

(c) Average performance over 60 synthetic data sets (local outliers and 10 percent global outliers).

Fig. 15. ROC AUC for local methods (LOF, LDOF, LoOP), global methods (*K*-NN and *K*-NN-weight), LOCI, and GLOSH on 180 synthetic data sets.

to the fixed cutoff value but take the complete information of the ranking into account. All reference algorithm implementations were taken from ELKI [Achtert et al. 2013].

Experimental Settings. All compared methods are using some neighborhood to compute the outlier model. We compare the results of different methods using different values of this common parameter, testing choices for the neighborhood size from 4 to 40 in steps of 4. For LOCI, we choose the parameter $r_{\max}$ depending on the datasets, as proposed by Papadimitriou et al. [2003], to allow a complete search between local and global ranges of the neighborhood radii. Accordingly, in the plots, we show the behavior of all methods except LOCI for the different values of the neighborhood size, and plot a straight line for LOCI (aLOCI) for the largest neighborhood size, which reflects the performance of assigning the maximum score of all neighborhood sizes up to this maximum.

Outlier Detection Results. On the synthetic datasets, we see the different levels of performance in particular for the local methods (LOF, LoOP, LDOF) when adding global outliers. In the datasets without global outliers (Figure 15(a)), the local methods and GLOSH are performing better than the global methods (*K*-NN and *K*-NN-weight). Both types perform better than LOCI. While, in these cases, the local methods deteriorate for the more noisy datasets with the increasing level of global outliers (Figures 15(b) and 15(c)) and the global methods (and LOCI) are performing increasingly well, GLOSH performs consistently well and is competitive then with the global methods. For most parameter settings, GLOSH is on top. This demonstrates the ability of GLOSH to identify local and global outliers equally well while the other methods are specialized either to local or to global outliers. The effect that we observe for LOCI, not being specialized to local or global outliers, might be that the task becomes easier for this method overall with more global outliers.

In the real datasets, it is not known *a priori* whether there are local outliers, global outliers, or outliers of both types. In the Lymphography dataset (Figure 16(a)), we observe no particular characteristic, all methods are performing more or less well, depending on different parameter values. We see, however, that GLOSH is not equally depending on the parameter values, but is more stable over the whole range of tested values. In most other cases (Figures 16(b)–16(e)), the global methods perform better than the local methods (in different degrees) but GLOSH is always in the top group, except for the Satellite Image 2 data (Figure 16(c)), in which GLOSH catches up with a larger neighborhood[11] while the local methods deteriorate. LOCI is performing best, compared to the other methods, on Satellite Image 2 (Figure 16(c)) and Waveform 2 (Figure 16(d)), and worst, being outperformed by all the other methods, on Pendigits 2 (Figure 16(b)), in which the LOCI result, with a ROC AUC value of approximately 0.5, is not different from a random ranking. On the other datasets, the performance of LOCI (aLOCI on KDD Cup 1999) is around the average of the other methods.

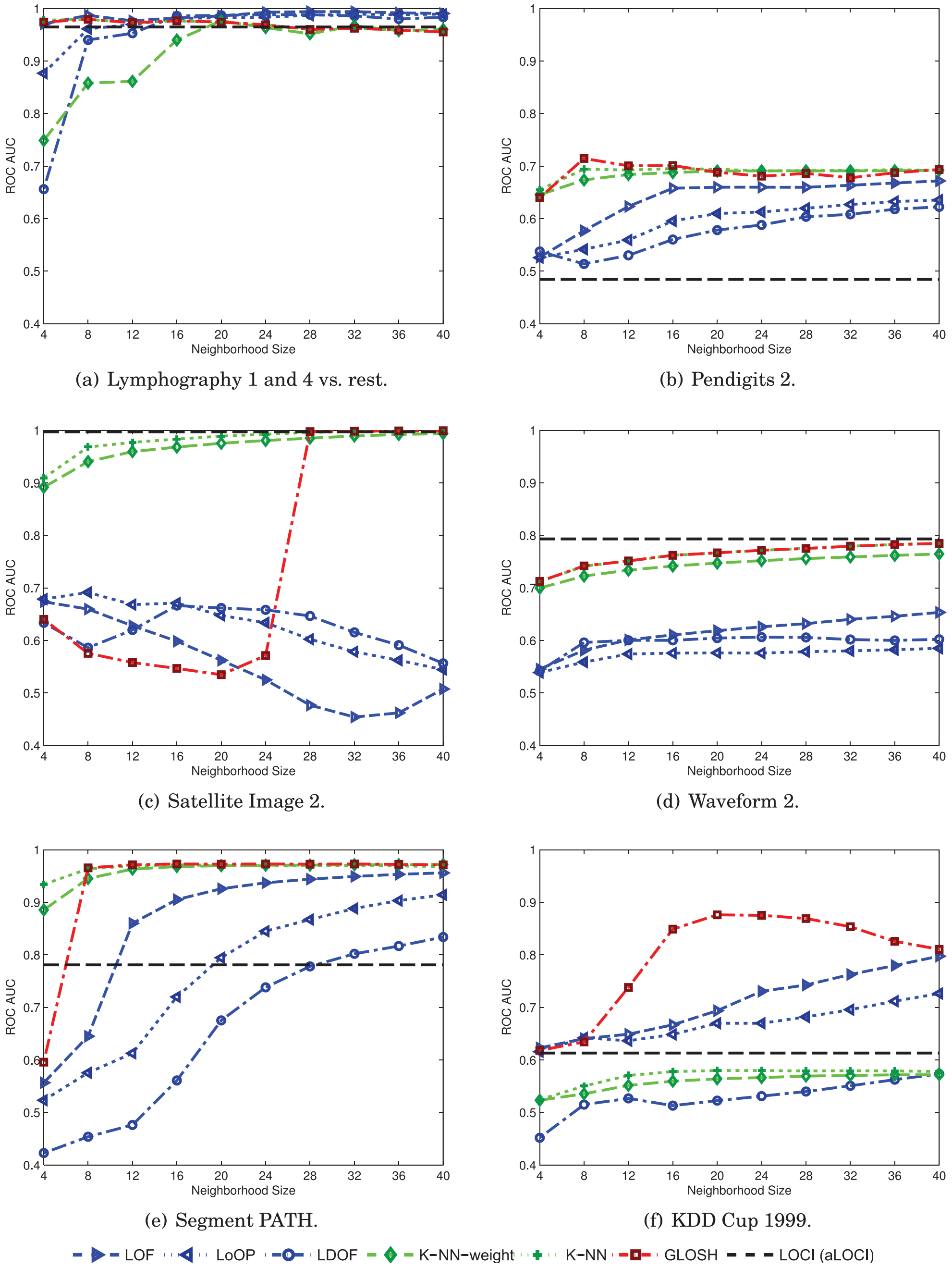
Interestingly, on most datasets, we can identify the families of local approaches and global approaches, as these follow a common family pattern, respectively. GLOSH and LOCI, by their performance pattern, do not fall into any of these families. However, GLOSH is typically in the top group while LOCI is sometimes best and sometimes worst. A particular case is the KDD Cup 1999 dataset (Figure 16(f)). Here, the local methods perform differently well, LOF and LoOP being better and LDOF being worse (actually around random performance) than the global methods, the performance of aLOCI being in between. GLOSH is on top also here, for some neighborhood sizes even with a large gap.

Overall, these results show that GLOSH, although not always outperforming other methods, can adapt quite well to different challenges. Where either global or local methods would be in an advantage, GLOSH can keep up with the top group in either scenario.

9. FINAL REMARKS

In this article, we proposed and discussed in detail a general framework for unsupervised and semisupervised data analysis, based on hierarchical density estimates. As a conceptually sound basis for density-based clustering and outlier detection, we described the hierarchical clustering algorithm HDBSCAN*, which produces a complete density-based clustering hierarchy representing all possible DBSCAN-like solutions for an infinite range of density thresholds. We showed that this cluster hierarchy can be processed in different ways to yield effective tools for cluster analysis as well as outlier detection.

[11] In this dataset, smaller neighborhoods and minimum cluster sizes allow for very small subcomponents with less than 30 objects (in contrast to almost 6000 objects in the whole data) to get split right at the top of the hierarchy. Neighborhood and minimum cluster sizes larger than 24 smoothen the hierarchy and prevent these spurious clusters to arise, keeping only the most prominent clusters and improving the outlier scores.



(a) Lymphography 1 and 4 vs. rest.

(b) Pendigits 2.

(c) Satellite Image 2.

(d) Waveform 2.

(e) Segment PATH.

(f) KDD Cup 1999.

Fig. 16. ROC AUC for local methods (LOF, LDOF, LoOP), global methods (*K*-NN and *K*-NN-weight), LOCI (aLOCI for KDD Cup 1999), and GLOSH on different real data sets.

For cluster analysis, we demonstrated how the HDBSCAN* hierarchy can be used to facilitate different types of visualizations, how it can be transformed into a simplified tree of only significant clusters, and how a flat partition composed of clusters extracted from local cuts through the cluster tree can be extracted that optimize certain cluster quality measures. We described an unsupervised measure of cluster stability for the purpose of cluster extraction, and we also showed how to integrate instance level constraints into the optimization procedure to allow cluster extraction in a semisupervised way.

For outlier detection, we defined a novel outlier measure called GLOSH, which can identify both global and local types of outliers. GLOSH is based on the density estimates available when the HDBSCAN* hierarchy is constructed, lending itself to an efficient algorithm for computing these scores, which we also see as bridging the gap towards reconciling the statistical and the database-oriented approaches for outlier detection.

In an extensive set of experiments, on a wide variety of synthetic and real world datasets, we have demonstrated that our methods for clustering and outlier detection perform, in general, much better and more robust than state-of-the-art methods.

There are several avenues for future research in the context of our proposed framework. The instance described in this article is based on a particular (albeit very commonly used) K-NN density estimate, which has certain limitations. Other types of density estimates should be investigated that can improve different aspects of the framework. One aspect is the quality of the results in certain scenarios, which can possibly be improved by using better estimates of the density. The other aspect is the runtime of the HDBSCAN* algorithm by using faster-to-compute estimates. To speedup the HDBSCAN* algorithm, one can also investigate how HDBSCAN* can be based on Cluster Features [Zhang et al. 1996] or Data Bubbles [Breunig et al. 2001], which have been successfully used to speedup other clustering algorithms. Another promising direction for further research is the investigation of alternative measures for cluster quality in the unsupervised and semisupervised scenarios. The relative excess of mass adopted in this article is by no means the only possible measure for cluster stability that can be used in our framework, and other kinds of constraints, other than should-link and should-not-link, may lend themselves to different types of quality measures for cluster extraction.

REFERENCES

N. Abe, B. Zadrozny, and J. Langford. 2006. Outlier detection by active learning. In *Proceedings of the 12th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD).*Philadelphia, PA. 504–509. DOI:http://dx.doi.org/10.1145/1150402.1150459

E. Achtert, H.-P. Kriegel, E. Schubert, and A. Zimek. 2013. Interactive data mining with 3D-parallel-coordinate-trees. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD).* New York City, NY. 1009–1012. DOI:http://dx.doi.org/10.1145/2463676.2463696

C. C. Aggarwal and P. S. Yu. 2001. Outlier detection for high dimensional data. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD).* Santa Barbara, CA. 37–46. DOI:http://dx.doi.org/10.1145/375663.375668

M. Agyemang, K. Barker, and R. Alhajj. 2006. A comprehensive survey of numeric and symbolic outlier mining techniques. *Intelligent Data Analysis* 10 (2006), 521–538.

F. Angiulli and F. Fassetti. 2009. DOLPHIN: An efficient algorithm for mining distance-based outliers in very large datasets. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 3, 1 (2009), 4:1–57. DOI:http://dx.doi.org/10.1145/1497577.1497581

F. Angiulli and C. Pizzuti. 2002. Fast outlier detection in high dimensional spaces. In *Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD).* Helsinki, Finland. 15–26. DOI:http://dx.doi.org/10.1007/3-540-45681-3_2

M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. 1999. OPTICS: Ordering points to identify the clustering structure. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD).* Philadelphia, PA. 49–60. DOI:http://dx.doi.org/10.1145/304182.304187.

K. Bache and M. Lichman. 2013. UCI Machine Learning Repository. (2013). http://archive.ics.uci.edu/ml.

V. Barnett. 1978. The study of outliers: Purpose and model. *Applied Statistics* 27, 3 (1978), 242–250.

V. Barnett and T. Lewis. 1994. *Outliers in Statistical Data* (3rd ed.). John Wiley & Sons.

S. Basu, I. Davidson, and K. Wagstaff (Eds.). 2008. *Constraint Clustering: Advances in Algorithms, Applications and Theory*. CRC Press, Boca Raton, London, New York.

S. D. Bay and M. Schwabacher. 2003. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *Proceedings of the 9th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. Washington, DC. 29–38. DOI:http://dx.doi.org/10.1145/956750.956758

R. J. Beckman and R. D. Cook. 1983. Outlier..........s. *Technometrics* 25, 2 (1983), 119–149.

M. Bilenko, S. Basu, and R. J. Mooney. 2004. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the 21st International Conference on Machine Learning (ICML)*. Banff, AB, Canada.

C. M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.

S. Brecheisen, H.-P. Kriegel, P. Kröger, and M. Pfeifle. 2004. Visually mining through cluster hierarchies. In *Proceedings of the 4th SIAM International Conference on Data Mining (SDM)*. Lake Buena Vista, FL. 400–412.

M. M. Breunig, H.-P. Kriegel, P. Kröger, and J. Sander. 2001. Data bubbles: Quality preserving performance boosting for hierarchical clustering. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*. Santa Barbara, CA. 79–90. DOI:http://dx.doi.org/10.1145/375663.375672

M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. 2000. LOF: Identifying density-based local outliers. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*. Dallas, TX. 93–104. DOI:http://dx.doi.org/10.1145/342009.335388

C. Böhm and C. Plant. 2008. HISSCLU: A hierarchical density-based method for semi-supervised clustering. In *Proceedings of the 11th International Conference on Extending Database Technology (EDBT)*. Nantes, France. 440–451. DOI:http://dx.doi.org/10.1145/1353343.1353398

R. J. G. B. Campello, D. Moulavi, and J. Sander. 2013a. Density-based clustering based on hierarchical density estimates. In *Proceedings of the 17th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*. Gold Coast, Australia. 160–172. DOI:http://dx.doi.org/10.1007/978-3-642-37456-2_14

R. J. G. B. Campello, D. Moulavi, A. Zimek, and J. Sander. 2013b. A framework for semi-supervised and unsupervised optimal extraction of clusters from hierarchies. *Data Mining and Knowledge Discovery* 27, 3 (2013), 344–371. DOI:http://dx.doi.org/10.1007/s10618-013-0311-4

M. H. Chehreghani, H. Abolhassani, and M. H. Chehreghani. 2008. Improving density-based methods for hierarchical clustering of web pages. *Data & Knowledge Engineering* 67, 1 (2008), 30–50. DOI:http://dx.doi.org/10.1016/j.datak.2008.06.006

D. Comaniciu and P. Meer. 1999. Distribution free decomposition of multivariate data. *Pattern Analysis and Applications* 2, 1 (1999), 22–30.

D. Coomans and D. L. Massart. 1981. Potential methods in pattern recognition: Part 2. CLUPOT – An unsupervised pattern recognition technique. *Analytica Chimica Acta* 133, 3 (1981), 225–239.

A. Cuevas, M. Febrero, and R. Fraiman. 2000. Estimating the number of clusters. *Canadian Journal of Statistics* 28, 2 (2000), 367–382.

A. Cuevas, M. Febrero, and R. Fraiman. 2001. Cluster analysis: A further approach based on density estimation. *Computational Statistics and Data Analysis* 36 (2001), 441–459. DOI:http://dx.doi.org/10.1016/j.csda.2005.10.012

X. H. Dang, I. Assent, R. T. Ng, A. Zimek, and E. Schubert. 2014. Discriminative features for identifying and interpreting outliers. In *Proceedings of the 30th International Conference on Data Engineering (ICDE), Chicago, IL*. 88–99. DOI:http://dx.doi.org/10.1109/ICDE.2014.6816642

M. Daszykowski, B. Walczak, and D. L. Massart. 2001. Looking for natural patterns in data: Part 1. Density-based approach. *Chemometrics and Intelligent Laboratory Systems* 56, 2 (2001), 83–92.

I. Davidson, K. L. Wagstaff, and S. Basu. 2006. Measuring constraint-set utility for partitional clustering algorithms. In *Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*. Berlin, Germany. 115–126.

T. de Vries, S. Chawla, and M. E. Houle. 2010. Finding local anomalies in very high dimensional space. In *Proceedings of the 10th IEEE International Conference on Data Mining (ICDM)*. Sydney, Australia. 128–137. DOI:http://dx.doi.org/10.1109/ICDM.2010.151

T. de Vries, S. Chawla, and M. E. Houle. 2012. Density-preserving projections for large-scale local anomaly detection. *Knowledge and Information Systems (KAIS)* 32, 1 (2012), 25–52. DOI:http://dx.doi.org/10.1007/s10115-011-0430-4

S. Dhandapani, G. Gupta, and J. Ghosh. 2010. *Design and Implementation of Scalable Hierarchical Density Based Clustering*. Technical Report. Technical Report IDEAL-2010-06, Dept. Electrical and Computer Eng., Univ. of Texas at Austin.

L. Ertöz, M. Steinbach, and V. Kumar. 2002. A new shared nearest neighbor clustering algorithm and its applications. In *Workshop on Clustering High Dimensional Data and its Applications at 2nd SIAM International Conference on Data Mining*. 105–115.

L. Ertöz, M. Steinbach, and V. Kumar. 2003. Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. In *Proceedings of the 3rd SIAM International Conference on Data Mining (SDM), San Francisco, CA*.

M. Ester. 2009. Density-based Clustering. In *Encyclopedia of Database Systems*, L. Liu and M. T. Özsu (Eds.). Springer, 795–799. DOI:http://dx.doi.org/10.1007/978-0-387-39940-9_605

M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd ACM International Conference on Knowledge Discovery and Data Mining (KDD)*. Portland, OR. 226–231.

B. S. Everitt, S. Landau, and M. Leese. 2001. *Cluster Analysis* (4th ed.). Arnold.

A. Foss and O. R. Zaïane. 2002. A parameterless method for efficiently discovering clusters of arbitrary shape in large datasets. In *Proceedings of the 2nd IEEE International Conference on Data Mining (ICDM), Maebashi City.* Japan. 179–186.

A. L. N. Fred and A. K. Jain. 2005. Combining multiple clusterings using evidence accumulation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27, 6 (2005), 835–850. DOI:http://dx.doi.org/10.1109/TPAMI.2005.113

K. Fukunaga and L. Hostetler. 1975. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory* 21, 1 (1975), 32–40.

J. Gao and P.-N. Tan. 2006. Converting output scores from outlier detection algorithms into probability estimates. In *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM)*. Hong Kong, China. 212–221. DOI:http://dx.doi.org/10.1109/ICDM.2006.43

J. M. Geusebroek, G. J. Burghouts, and A. W. M. Smeulders. 2005. The amsterdam library of object images. *International Journal of Computer Vision* 61, 1 (2005), 103–112. DOI:http://dx.doi.org/10.1023/B:VISI.0000042993.50813.60

F. E. Grubbs. 1950. Sample criteria for testing outlying observations. *The Annals of Mathematical Statistics* 21, 1 (1950), 27–58.

G. Gupta, A. Liu, and J. Ghosh. 2006. Hierarchical density shaving: A clustering and visualization framework for large biological datasets. In *IEEE ICDM Workshop on Data Mining in Bioinformatics (DMB)*. Hong Kong, China, 89–93.

G. Gupta, A. Liu, and J. Ghosh. 2010. Automated hierarchical density shaving: A robust automated clustering and visualization framework for large biological data sets. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 7, 2 (2010), 223–237. DOI:http://dx.doi.org/10.1145/1791396.1791402

A. S. Hadi, A. H. M. Rahmatullah Imon, and M. Werner. 2009. Detection of outliers. *Wiley Interdisciplinary Reviews: Computational Statistics* 1, 1 (2009), 57–70.

S. Hang, Z. You, and L. Y. Chun. 2009. Incorporating biological knowledge into density-based clustering analysis of gene expression data. In *6th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, Vol. 5. Tianjin, China, 52–56.

J. A. Hanley and B. J. McNeil. 1982. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology* 143 (1982), 29–36.

J. A. Hartigan. 1975. *Clustering Algorithms*. John Wiley & Sons, New York, London, Sydney, Toronto.

J. A. Hartigan. 1987. Estimation of a convex density contour in two dimensions. *J. Amer. Statist. Assoc.* 82, 397 (1987), 267–270.

D. Hawkins. 1980. *Identification of Outliers*. Chapman and Hall.

M. Herbin, N. Bonnet, and P. Vautrot. 2001. Estimation of the number of clusters and influence zones. *Pattern Recognition Letters* 22, 14 (2001), 1557–1568. DOI:http://dx.doi.org/10.1016/S0167-8655(01)00103-9

A. Hinneburg and H. H. Gabriel. 2007. Denclue 2.0: Fast clustering based on kernel density estimation. In *Proceedings of the 7th International Symposium on Intelligent Data Analysis (IDA).* Ljubljana, Slovenia. 70–80. DOI:http://dx.doi.org/10.1007/978-3-540-74825-0_7

A. Hinneburg and D. A. Keim. 1998. An efficient approach to clustering in large multimedia databases with noise. In *Proceedings of the 4th ACM International Conference on Knowledge Discovery and Data Mining (KDD).* New York City, NY. 58–65.

A. Hinneburg and D. A. Keim. 2003. A general approach to clustering in large databases with noise. *Knowledge and Information Systems (KAIS)* 5 (2003), 387–415. Issue 4.

V. J. Hodge and J. Austin. 2004. A survey of outlier detection methodologies. *Artificial Intelligence Review* 22 (2004), 85–126. DOI:http://dx.doi.org/10.1023/B:AIRE.0000045502.10941.a9

D. Horta and R. J. G. B. Campello. 2012. Automatic aspect discrimination in data clustering. *Pattern Recognition* 45, 12 (2012), 4370–4388. DOI:http://dx.doi.org/10.1016/j.patcog.2012.05.011

L. Hubert and P. Arabie. 1985. Comparing partitions. *Journal of Classification* 2, 1 (1985), 193–218.

J.-N. Hwang, S.-R. Lay, and A. Lippman. 1994. Nonparametric multivariate density estimation: A comparative study. *IEEE Transactions on Signal Processing* 42, 10 (1994), 2795–2810.

A. K. Jain and R. C. Dubes. 1988. *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs.

W. Jin, A. K. Tung, and J. Han. 2001. Mining top-n local outliers in large databases. In *Proceedings of the 7th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD).* San Francisco, CA. 293–298. DOI:http://dx.doi.org/10.1145/502512.502554

W. Jin, A. K. H. Tung, J. Han, and W. Wang. 2006. Ranking outliers using symmetric neighborhood relationship. In *Proceedings of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD).* Singapore. 577–593. DOI:http://dx.doi.org/10.1007/11731139_68

S. C. Johnson. 1967. Hierarchical clustering schemes. *Psychometrika* 32 (1967), 241–254.

F. Keller, E. Müller, and K. Böhm. 2012. HiCS: High contrast subspaces for density-based outlier ranking. In *Proceedings of the 28th International Conference on Data Engineering (ICDE).* Washington, DC. DOI:http://dx.doi.org/10.1109/ICDE.2012.88

E. M. Knorr and R. T. Ng. 1997a. A unified approach for mining outliers. In *Proceedings of the conference of the Centre for Advanced Studies on Collaborative research (CASCON).* Toronto, ON, Canada. 11–23. DOI:http://dx.doi.org/10.1145/782010.782021

E. M. Knorr and R. T. Ng. 1997b. A unified notion of outliers: Properties and computation. In *Proceedings of the 3rd ACM International Conference on Knowledge Discovery and Data Mining (KDD).* Newport Beach, CA. 219–222. DOI:http://dx.doi.org/10.1145/782010.782021

E. M. Knorr and R. T. Ng. 1998. Algorithms for mining distance-based outliers in large datasets. In *Proceedings of the 24th International Conference on Very Large Data Bases (VLDB).* New York City, NY. 392–403.

E. M. Knorr, R. T. Ng, and V. Tucanov. 2000. Distance-based outliers: Algorithms and applications. *The VLDB Journal* 8, 3–4 (2000), 237–253. DOI:http://dx.doi.org/10.1007/s007780050006

G. Kollios, D. Gunopulos, N. Koudas, and S. Berchthold. 2003. Efficient biased sampling for approximate clustering and outlier detection in large datasets. *IEEE Transactions on Knowledge and Data Engineering* 15, 5 (2003), 1170–1187. DOI:http://dx.doi.org/10.1109/TKDE.2003.1232271

H.-P. Kriegel, P. Kröger, J. Sander, and A. Zimek. 2011. Density-based clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 1, 3 (2011), 231–240. DOI:http://dx.doi.org/10.1002/widm.30

H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. 2009a. LoOP: Local outlier probabilities. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM).* Hong Kong, China. 1649–1652. DOI:http://dx.doi.org/10.1145/1645953.1646195

H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. 2009b. Outlier detection in axis-parallel subspaces of high dimensional data. In *Proceedings of the 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD).* Bangkok, Thailand. 831–838. DOI:http://dx.doi.org/10.1007/978-3-642-01307-2_86

H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. 2011a. Interpreting and unifying outlier scores. In *Proceedings of the 11th SIAM International Conference on Data Mining (SDM)* Mesa, AZ. 13–24.

H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. 2012. Outlier detection in arbitrarily oriented subspaces. In *Proceedings of the 12th IEEE International Conference on Data Mining (ICDM).* Brussels, Belgium. 379–388. DOI:http://dx.doi.org/10.1109/ICDM.2012.21

H.-P. Kriegel, P. Kröger, and A. Zimek. 2010. Outlier Detection Techniques. Tutorial at the 16th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD). Washington, DC. (2010).

H.-P. Kriegel, M. Schubert, and A. Zimek. 2008. Angle-based outlier detection in high-dimensional data. In *Proceedings of the 14th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD).* Las Vegas, NV. 444–452. DOI:http://dx.doi.org/10.1145/1401890.1401946

B. Larsen and C. Aone. 1999. Fast and effective text mining using linear-time document clustering. In *Proceedings of the 5th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD).* San Diego, CA. 16–22.

A. Lazarevic and V. Kumar. 2005. Feature bagging for outlier detection. In *Proceedings of the 11th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD).* Chicago, IL. 157–166. DOI:http://dx.doi.org/10.1145/1081870.1081891

L. Lelis and J. Sander. 2009. Semi-supervised density-based clustering. In *Proceedings of the 9th IEEE International Conference on Data Mining (ICDM).* Miami, FL. 842–847. DOI:http://dx.doi.org/10.1109/ICDM.2009.143

F. T. Liu, K. M. Ting, and Z.-H. Zhou. 2012. Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 6, 1 (2012), 3:1–39.

P. Liu, D. Zhou, and N. Wu. 2007. VDBSCAN: Varied density based spatial clustering of applications with noise. In *International Conference on Service Systems and Service Management (ICSSSM).* Chengdu, China. 1–4.

B. Micenková, R. T. Ng, X. H. Dang, and I. Assent. 2013. Explaining outliers by subspace separability. In *Proceedings of the 13th IEEE International Conference on Data Mining (ICDM).* Dallas, TX. 518–527.

D. W. Muller and G. Sawitzki. 1991. Excess mass estimates and tests for multimodality. *J. Amer. Statist. Assoc.* 86, 415 (1991), 738–746.

E. Müller, I. Assent, P. Iglesias, Y. Mülle, and K. Böhm. 2012. Outlier ranking via subspace analysis in multiple views of the data. In *Proceedings of the 12th IEEE International Conference on Data Mining (ICDM).* Brussels, Belgium. 529–538. DOI:http://dx.doi.org/10.1109/ICDM.2012.112

E. Müller, I. Assent, U. Steinhausen, and T. Seidl. 2008. OutRank: Ranking outliers in high dimensional data. In *Proceedings of the 24th International Conference on Data Engineering (ICDE) Workshop on Ranking in Databases (DBRank).* Cancun, Mexico. 600–603. DOI:http://dx.doi.org/10.1109/ICDEW.2008.4498387

E. Müller, M. Schiffer, and T. Seidl. 2010. Adaptive outlierness for subspace outlier ranking. In *Proceedings of the 19th ACM Conference on Information and Knowledge Management (CIKM).* Toronto, ON, Canada. 1629–1632. DOI:http://dx.doi.org/10.1145/1871437.1871690

E. Müller, M. Schiffer, and T. Seidl. 2011. Statistical selection of relevant subspace projections for outlier ranking. In *Proceedings of the 27th International Conference on Data Engineering (ICDE).* Hannover, Germany. 434–445. DOI:http://dx.doi.org/10.1109/ICDE.2011.5767916

M. C. Naldi, R. J. G. B. Campello, E. R. Hruschka, and A. C. P. L. F. Carvalho. 2011. Efficiency issues of evolutionary k-means. *Applied Soft Computing* 11, 2 (2011), 1938–1952.

H. V. Nguyen, H. H. Ang, and V. Gopalkrishnan. 2010. Mining outliers with ensemble of heterogeneous detectors on random subspaces. In *Proceedings of the 15th International Conference on Database Systems for Advanced Applications (DASFAA).* Tsukuba, Japan. 368–383. DOI:http://dx.doi.org/10.1007/978-3-642-12026-8_29

H. V. Nguyen and V. Gopalkrishnan. 2009. Efficient pruning schemes for distance-based outlier detection. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD).* Bled, Slovenia. 160–175. DOI:http://dx.doi.org/10.1007/978-3-642-04174-7_11

H. V. Nguyen, V. Gopalkrishnan, and I. Assent. 2011. An unbiased distance-based outlier detection approach for high-dimensional data. In *Proceedings of the 16th International Conference on Database Systems for Advanced Applications (DASFAA).* Hong Kong, China. 138–152. DOI:http://dx.doi.org/10.1007/978-3-642-20149-3_12

G. H. Orair, C. Teixeira, Y. Wang, W. Meira Jr., and S. Parthasarathy. 2010. Distance-based outlier detection: Consolidation and renewed bearing. *Proceedings of the VLDB Endowment* 3, 2 (2010), 1469–1480.

S. Papadimitriou, H. Kitagawa, P. B. Gibbons, and C. Faloutsos. 2003. LOCI: Fast outlier detection using the local correlation integral. In *Proceedings of the 19th International Conference on Data Engineering (ICDE).* Bangalore, India. 315–326. DOI:http://dx.doi.org/10.1109/ICDE.2003.1260802

D. Pascual, F. Pla, and J. S. Sánchez. 2006. Non parametric local density-based clustering for multimodal overlapping distributions. In *Proceedings of the 7th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL).* Burgos, Spain. 671–678.

E. S. Pearson and C. Chandra Sekar. 1936. The efficiency of statistical tools and a criterion for the rejection of outlying observations. *Biometrika* 28, 3/4 (1936), 308–320.

T. Pei, A. Jasra, D. J. Hand, A.-X. Zhu, and C. Zhou. 2009. DECODE: A new method for discovering clusters of different densities in spatial data. *Data Mining and Knowledge Discovery* 18, 3 (2009), 337–369. DOI:http://dx.doi.org/10.1007/s10618-008-0120-3

T. Pei, A.-X. Zhu, C. Zhou, B. Li, and C. Qin. 2006. A new approach to the nearest-neighbour method to discover cluster features in overlaid spatial point processes. *International Journal of Geographical Information Science* 20, 2 (2006), 153–168.

N. Pham and R. Pagh. 2012. A near-linear time approximation algorithm for angle-based outlier detection in high-dimensional data. In *Proceedings of the 18th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. Beijing, China.

S. Ramaswamy, R. Rastogi, and K. Shim. 2000. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD).* Dallas, TX. 427–438. DOI:http://dx.doi.org/10.1145/342009.335437

C. Ruiz, M. Spiliopoulou, and E. Menasalvas. 2007. C-DBSCAN: Density-based clustering with constraints. In *Rough Sets, Fuzzy Sets, Data Mining and Granular Computing*, A. An, J. Stefanowski, S. Ramanna, C. Butz, W. Pedrycz, and G. Wang (Eds.). 216–223.

C. Ruiz, M. Spiliopoulou, and E. Menasalvas. 2010. Density-based semi-supervised clustering. *Data Mining and Knowledge Discovery* 21, 3 (2010), 345–370.

J. Sander. 2010. Density-based clustering. In *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb (Eds.). Springer, US, 270–273.

J. Sander, X. Qin, Z. Lu, N. Niu, and A. Kovarsky. 2003. Automatic extraction of clusters from hierarchical clustering representations. In *Proceedings of the 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*. Seoul, Korea. 75–87. DOI:http://dx.doi.org/10.1007/3-540-36175-8_8

E. Schubert, R. Wojdanowski, A. Zimek, and H.-P. Kriegel. 2012. On evaluation of outlier rankings and outlier scores. In *Proceedings of the 12th SIAM International Conference on Data Mining (SDM)*. Anaheim, CA. 1047–1058.

E. Schubert, A. Zimek, and H.-P. Kriegel. 2014a. Generalized outlier detection with flexible kernel density estimates. In *Proceedings of the 14th SIAM International Conference on Data Mining (SDM)*. Philadelphia, PA. 542–550. DOI:http://dx.doi.org/10.1137/1.9781611973440.63

E. Schubert, A. Zimek, and H.-P. Kriegel. 2014b. Local outlier detection reconsidered: A generalized view on locality with applications to spatial, video, and network outlier detection. *Data Mining and Knowledge Discovery* 28, 1 (2014), 190–237. DOI:http://dx.doi.org/10.1007/s10618-012-0300-z

B. W. Silverman. 1986. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC.

P. H. A. Sneath. 1957. The application of computers to taxonomy. *Journal of General Microbiology* 17 (1957), 201–226.

T. Soler and M. Chin. 1985. On transformation of covariance matrices between local Cartesian coordinate systems and commutative diagrams. In *ASP-ACSM Convention*. 393–406.

W. Stuetzle. 2003. Estimating the cluster tree of a density by analyzing the minimal spanning tree of a sample. *Journal of Classification* 20, 1 (2003), 25–47. DOI:http://dx.doi.org/10.1007/s00357-003-0004-6

W. Stuetzle and R. Nugent. 2010. A generalized single linkage method for estimating the cluster tree of a density. *Journal of Computational and Graphical Statistics* 19, 2 (2010), 397–418.

H. Sun, J. Huang, J. Han, H. Deng, P. Zhao, and B. Feng. 2010. gSkeletonClu: Density-based network clustering via structure-connected tree division or agglomeration. In *Proceedings of the 10th IEEE International Conference on Data Mining (ICDM)*. Sydney, Australia. 481–490.

P.-N. Tan, M. Steinbach, and V. Kumar. 2006. *Introduction to Data Mining*. Addison Wesley.

J. Tang, Z. Chen, A. W.-C. Fu, and D. W. Cheung. 2002. Enhancing effectiveness of outlier detections for low density patterns. In *Proceedings of the 6th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*. Taipei, Taiwan. 535–548. DOI:http://dx.doi.org/10.1007/3-540-47887-6_53

K. L. Wagstaff. 2002. *Intelligent Clustering with Instance-Level Constraints*. Ph.D. Dissertation. Department of Computer Science, Cornell University.

D. Wishart. 1969. Mode analysis: A generalization of nearest neighbor which reduces chaining effects. In *Numerical Taxonomy*, A. J. Cole (Ed.). 282–311.

M. A. Wong and T. Lane. 1983. A kth nearest neighbour clustering procedure. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 45, 3 (1983), 362–368.

R. Xu and D. Wunsch II. 2005. Survey of clustering algorithms. *IEEE Transactions on Neural Networks* 16 (2005), 645–678. DOI:http://dx.doi.org/10.1109/TNN.2005.845141

R. Xu and D. Wunsch II. 2009. *Clustering*. IEEE Press.

J. Yang, N. Zhong, Y. Yao, and J. Wang. 2008. Local peculiarity factor and its application in outlier detection. In *Proceedings of the 14th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. Las Vegas, NV. 776–784. DOI:http://dx.doi.org/10.1145/1401890.1401983

K. Y. Yeung, C. Fraley, A. Murua, A. E. Raftery, and W. L. Ruzzo. 2001. Model-based clustering and data transformations for gene expression data. *Bioinformatics* 17, 10 (2001), 977–987.

K. Y. Yeung, M. Medvedovic, and R. E. Bumgarner. 2003. Clustering gene-expression data with repeated measurements. *Genome Biology* 4, 5 (2003), R34.1–R34.17.

K. Zhang, M. Hutter, and H. Jin. 2009. A new local distance-based outlier detection approach for scattered real-world data. In *Proceedings of the 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*. Bangkok, Thailand. 813–822. DOI:http://dx.doi.org/10.1007/978-3-642-01307-2_84

T. Zhang, R. Ramakrishnan, and M. Livny. 1996. BIRCH: An efficient data clustering method for very large databases. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*. Montreal, QC, Canada. 103–114. DOI:http://dx.doi.org/10.1145/233269.233324

A. Zimek, R. J. G. B. Campello, and J. Sander. 2013. Ensembles for unsupervised outlier detection: Challenges and research questions. *ACM SIGKDD Explorations* 15, 1 (2013), 11–22.

A. Zimek, R. J. G. B. Campello, and J. Sander. 2014. Data perturbation for outlier detection ensembles. In *Proceedings of the 26th International Conference on Scientific and Statistical Database Management (SSDBM)*. Aalborg, Denmark. 13:1–12. DOI:http://dx.doi.org/10.1145/2618243.2618257

A. Zimek, M. Gaudet, R. J. G. B. Campello, and J. Sander. 2013a. Subsampling for efficient and effective unsupervised outlier detection ensembles. In *Proceedings of the 19th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. Chicago, IL. 428–436. DOI:http://dx.doi.org/10.1145/2487575.2487676

A. Zimek, E. Schubert, and H.-P. Kriegel. 2012. A survey on unsupervised outlier detection in high-dimensional numerical data. *Statistical Analysis and Data Mining* 5, 5 (2012), 363–387. DOI:http://dx.doi.org/10.1002/sam.11161

Received July 2013; revised December 2014; accepted February 2015