Top-Down Parameter-Free Clustering of High-Dimensional Categorical Data

Eugenio Cesario, Giuseppe Manco, and Riccardo Ortale

Abstract—A parameter-free, fully-automatic approach to clustering high-dimensional categorical data is proposed. The technique is based on a two-phase iterative procedure, which attempts to improve the overall quality of the whole partition. In the first phase, cluster assignments are given, and a new cluster is added to the partition by identifying and splitting a low-quality cluster. In the second phase, the number of clusters is fixed, and an attempt to optimize cluster assignments is done. On the basis of such features, the algorithm attempts to improve the overall quality of the whole partition and finds clusters in the data, whose number is naturally established on the basis of the inherent features of the underlying data set rather than being previously specified. Furthermore, the approach is parametric to the notion of cluster quality: Here, a cluster is defined as a set of tuples exhibiting a sort of homogeneity. We show how a suitable notion of cluster homogeneity can be defined in the context of high-dimensional categorical data, from which an effective instance of the proposed clustering scheme immediately follows. Experiments on both synthetic and real data prove that the devised algorithm scales linearly and achieves nearly optimal results in terms of compactness and separation.

Index Terms—Clustering, database applications, information search and retrieval.

1 Introduction

1.1 Motivations

Clustering is an unsupervised classification technique that aims at grouping a set of unlabeled objects into meaningful clusters [19], [25], with the requirement that the resulting groups are *homogeneous* (that is, pairs of objects in the same cluster are highly similar) and *neatly separated* (that is, objects within distinct clusters are very dissimilar). Clustering techniques have been extensively studied in several communities [1], [11], [20], [42]. Recently, increasing attention has been paid to clustering categorical data [2], [3], [14], [16], [21], [40], [41], where records are made up of nonnumerical data, since this task is of great practical relevance in several fields ranging from statistics to psychology [3].

There are a number of challenges in clustering categorical data. First, the lack of an inherent order on the domains of the individual attributes prevents the definition of a notion of similarity, which catches resemblance between categorical data objects. Clearly, this imposes difficulties at devising a suitable clustering quality that are not encountered in the case of numeric attributes, where, instead, object similarity naturally follows from the geometric properties of the data. Furthermore, categorical data is often high dimensional, thus requiring methods that are actually capable of scaling with dimensionality.

High-dimensional categorical data such as market-basket and Web usage data is a particular facet of categorical data. Records in such data sets include a large number of attributes, typically with Boolean values. Several emerging application settings require clustering techniques that provide an effective treatment of this kind of data, such as text analysis, bioinformatics, e-commerce, astronomy, and the insurance industry [1], [28], [33]. Unfortunately, most of the conventional approaches for categorical data do not properly scale to cluster volumes of high-dimensional data in terms of effectiveness and efficiency. Moreover, concerns related to data sparseness and/or skewness, as well as attribute irrelevancy and/or redundancy, typically impose looking for valuable clusters within several subsets of the original attribute space. This inevitably penalizes the effectiveness of clustering and further exacerbates its time requirements, since high-dimensional categorical data tends to exhibit different clusters on distinct attribute subsets.

In addition, the great majority of conventional clustering algorithms generally require an appropriate model selection strategy due to their dependency on multiple parameters, which may be difficult to tune. In its simplest form, the problem of model selection is concerned with the estimation of the "optimal" number of clusters. Although optimality can be difficult to pin down without some assumptions being made, some pragmatical cross-validation methods based on ad hoc quality criteria have been proposed in the literature. However, most clustering algorithms require the setting of many input parameters. Parameter-laden techniques are critical in several aspects [26]. First, incorrect settings may cause an algorithm to fail in finding the true patterns. Also, a perhaps more insidious problem is that the algorithm may report spurious patterns that do not really exist or greatly overestimate the significance of the reported patterns. This is especially likely when the user fails to understand the role of parameters.

Parameters are useful in a mining algorithm when they encode human-domain knowledge, thus allowing us to catch the user's explanation of the data under investigation. The

• *The authors are with the Institute of High Performance Computing and Networks (ICAR-CNR), Via Bucci 41c, I87036 Rende (CS), Italy. E-mail: {cesario, manco, ortale}@icar.cnr.it.*

Manuscript received 12 Apr. 2006; revised 26 Mar. 2007; accepted 17 July 2007; published online 8 Aug. 2007.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0170-0406.
Digital Object Identifier no. 10.1109/TKDE.2007.190649.

knowledge discovery process is highly interactive in its nature: The outcome of pattern discovery is a set of suggested hypotheses, which still require that they be further interpreted and tested according to the user's beliefs and presumptions. Under this perspective, parameters should be solely used to ease the embedding of application semantics within the pattern discovery task. Instead, parameters are often used for supporting different requirements such as efficiency, scalability, and flexibility. In this respect, they often introduce a bias within the algorithm, which makes the identification of the true patterns problematic. As a matter of fact, in many cases, parameters are tuned so that the clustering scheme is highly effective on data sets with certain properties while delivering poor performances with even small changes in the underlying data characteristics.

Thus, the effort in identifying a proper tuning for parameters may make it difficult to fully understand the effectiveness of a mining scheme. Indeed, this would conceptually require the investigation of the behavior of the algorithm on any data set across all possible parameter settings. Such an issue is even more evident when the performances of multiple schemes are compared, which is due to the requirement for intensively testing over as many parameter spaces.

Data mining algorithms should have as few such parameters as possible, ideally none. A parameter-free algorithm would limit the human ability to impose prejudices, expectations, and presumptions on the problem at hand and would let the data itself speak. These techniques are particularly useful when the data is described by several relevant attributes. As to the specific case of clustering, this latter aspect is even more relevant due to the intrinsic human inability of taking into account all possible meaningful groupings of data attributes.

It is true that the high interactive nature of the structure discovery process poses substantial limits to the effectiveness of a fully automatic approach. Typically, the latter involves strong assumptions about the underlying model, which clearly enforce the results. If the hypothesized model changes according to a user's beliefs and/or expectations, automatic methods may not work anymore, thus failing to catch the user's explanation of the data under investigation.

Notwithstanding, we believe that automatic techniques are complementary to human abilities. Indeed, a structure discovery method can help both in the first exploratory approach to data analysis and in developing further in-depth customized investigations on the resulting patterns. Indeed, it can effectively and rapidly search huge amounts of high-dimensional data in a way that is not possible for a human expert, whereas the latter can explain data and validate the results of autonomous methods in the light of her/his background knowledge. In this respect, automatic techniques are really useful, as long as they release the human expert from explicitly dealing with data-specific parameters (that is, those parameters that are inherently related to the properties of the data at hand). Possibly, as a reasonable trade-off between ease of use and flexibility, autonomous approaches can be enhanced to somehow embody human-domain knowledge by making them depend on a strict number of application-specific parameters, which permit different explanations of the data (according to the pursued task), if any.

1.2 Objectives and Contributions

The aforementioned problems have been tackled separately, and specific approaches have been proposed in the literature, which hardly fit the whole framework. The main objective of this paper is, instead, to face the three issues in a unified framework. We look forward to an algorithmic technique that is capable of automatically detecting the underlying interesting structure (when available) on high-dimensional categorical data. We are particularly interested in devising solutions that are scalable both in the size and in the dimensionality of the data and are simultaneously capable of adopting themselves to the patterns underlying the data.

In the following, we present *Automatic Top-Down Clustering* (AT-DC), a new approach to clustering high-dimensional categorical data, that scales to processing large volumes of such data in terms of both effectiveness and efficiency. The main idea of the approach is borrowed from the classical top-down approach to decision-tree learning, which recursively partitions the available data on the basis of the gain in purity of the subsets with respect to the original data set. AT-DC implements a similar strategy for clustering high-dimensional categorical data. Given an initial data set, it searches for a partition, which improves the overall purity. The algorithm is independent of any data-specific parameter (such as the number of clusters or occurrence thresholds for frequent attribute values). By contrast, it is deliberately left parametric to the notion of purity, which allows for adopting the quality criterion that best meets the specific applicative goal of clustering.

In this paper, we deal with a specific clustering requirement, that is, the discovery of syntactically homogeneous groups of categorical data. To this purpose, the frequency of attribute values is suitably employed to measure the extent of purity of the discovered clusters. This is a quite natural clustering model when dealing with categorical data. Notice that, in principle, the exploitation of a frequency-based quality criterion suffers from the same limitations that affect exact match similarity schemes. However, this aspect is not a concern at all, as long as syntactic similarity remains the required cluster property. In addition, we briefly discuss an idea for plugging a different quality criterion into the AT-DC framework, which would enable the identification of semantic similarity, another major clustering requirement, that typically arises when distinct values of an attribute are used for denoting the same concept.

AT-DC implements a parameter-free, fully-automatic approach. Starting from the basic assumption that clusters are groups of tuples exhibiting a high degree of overlap, the search for syntactically homogeneous clusters is exclusively guided by a quality criterion that is capable of evaluating the aforementioned property. In practice, the algorithm autonomously tries to improve the current partition by looking for a convenient cluster split, which isolates clusters of transactions with mostly frequent attribute values. AT-DC halts only when such a split is not available. This allows a true exploratory strategy, which does not impose any

presumptions on the data. In addition, the accuracy of our approach is comparable (actually, in most cases, even superior) to those of parameter-laden algorithms, even if we allow these algorithms to search exhaustively over their parameter spaces.

Interestingly, the AT-DC algorithmic scheme can be further customized, as it is decoupled from the underlying notion of quality, which governs the cluster generation process. In this paper, we instantiated it to clustering that are specific of high-dimensional categorical data. However, alternative instantiations can be obtained by adopting different measures, which encode specific application requirements.

1.3 Plan of the Paper

The rest of this paper is organized as follows: We start by critically reviewing in Section 2 the approaches available in the literature. Section 3 provides a detailed description of the AT-DC approach to clustering high-dimensional categorical data. The behavior of the algorithm, both on real-life and on synthesized data sets, is then analyzed in Section 4. Finally, Section 5 draws conclusions and highlights extensions to AT-DC, which are worth further research.

2 Related Works

Several approaches have been proposed in the current literature for clustering categorical data. However, the majority of these techniques typically suffer from two main limitations, namely, their dependency on a set of parameters that need to be properly tuned and their lack of scalability to large values of n and m. In particular, for this latter aspect, we remark that several effective approaches to clustering both categorical data and high-dimensional data exist. However, most of these approaches are inadequate both in dealing with the above features in a unified framework and in providing smart strategies for the correct estimation of parameters.

In principle, several distance-based clustering algorithms [10], [25], [31] can be adapted to transactional data. However, traditional clustering techniques are faced with the curse of dimensionality and the associated sparsity issues when dealing with very high-dimensional data such as market-basket data or Web sessions. For example, the K-Means algorithm has been adopted by replacing the cluster mean with the more robust notion of cluster medoid [31] (that is, the object within the cluster with the minimal distance from the other points) or the attribute mode [24]. However, the proposed extensions are inadequate for large values of m: Gozzi et al. [18] describe such inadequacies in detail and propose further extensions to the K-Means scheme, which fit transactional data. Unfortunately, this approach reveals to be parameter laden.

The point is that distance-based algorithms do not perform well when the number of dimensions is high. Indeed, several irrelevant attributes might distort the dissimilarity between tuples. Although standard dimension reduction techniques [9] can be used for detecting the relevant dimensions, these can be different for different clusters, thus invalidating such a preprocessing task. Recently, several clustering techniques have been proposed, which identify clusters in subspaces of maximum dimensionality (see [33] for a survey). Although most of these approaches were defined for numerical data, some recent works [13], [41] also consider subspace clustering for categorical data.

A different perspective in exploiting (dis)similarity is provided by the ROCK algorithm [21]. The core of the approach is an agglomerative hierarchical clustering procedure based on the concepts of *neighbors* and *links*. For a given tuple x, a tuple y is a neighbor of x if the Jaccard similarity $J(x, y)$ between them exceeds a prespecified threshold θ. Hence, the algorithm starts by assigning each tuple to a singleton cluster and merges clusters on the basis of the number of neighbors (links) that they share until the desired number of clusters is reached. ROCK is robust to high-dimensional data. However, the dependency of the algorithm to the parameter θ makes proper tuning difficult. This is further by the high computational complexity $(O(n^2 \log n))$ needed to compute links between objects.

Beyond the concept of (dis)similarity, clusters of categorical data can informally be understood as especially dense interval regions within the data set. The density notion is related to the frequency of specific groups of attribute values: The higher the frequency of such groups, the stronger the clustering. In a sense, this corresponds to preprocessing the data set by extracting relevant features (frequent patterns) and discovering clusters on the basis of these features. There are several approaches accounting for frequencies. As an example, Yang et al. [40] propose an approach based on histograms: The goodness of a cluster is higher if the average frequency of an item is high, as compared to the number of items appearing within a transaction. The algorithm is particularly suitable for large high-dimensional databases, but it is sensitive to a user-defined parameter (the *repulsion factor*), which weights the importance of the compactness/sparseness of a cluster. Other approaches [23], [32], [39], [41] extend the computation of frequencies to frequent patterns in the underlying data set. In particular, in [23], [32], each transaction is seen as a relation over some sets of items, and a hypergraph model is used for representing these relations. Hypergraph partitioning algorithms can hence be used for obtaining item/transaction clusters.

In general, approaches based on graph/hypergraph partitioning are common in the literature [14], [16], [41]. The key intuition here is to encode the data set into weighted summarization structures such as graphs, where individual attribute values correspond to weighted vertices. Thus, significant patterns are mapped into a hypergraph structure, and the exploitation of ad hoc hypergraph partitioning algorithms allows one to obtain a suitable clustering. The cost of clustering on the basis of such structures is acceptable, provided that the underlying data is low dimensional. Otherwise, one has to resort to abstractions, which reduce the dimensionality of the data.

Paradigmatic is the CLICKS algorithm, recently proposed in [41]. The algorithm encodes a data set into a weighted graph structure $G = (N, E)$, where the individual attribute values correspond to weighted vertices in N, and two nodes are connected by an edge if there is a tuple where

the corresponding attribute values co-occur. The algorithm starts from the observation that clusters correspond to dense (that is, with frequency higher than a user-specified threshold) maximal k-partite cliques and proceeds by enumerating all maximal k-partite cliques and checking their frequency. A crucial step is the computation of strongly connected components, that is, pairs of attribute values whose co-occurrence is above the specified threshold. For large values of m (or, more generally, when the number of dimensions or the cardinality of each dimension is high), this is an expensive task, which invalidates the efficiency of the approaches. In addition, technique depends upon a set of parameters, whose tuning can be problematic in practical cases. The experimental section in this paper describes such the inherent difficulties in detail.

Categorical clustering can also be tackled by using information-theoretic principles and the notion of entropy to measure closeness between objects. The basic intuition is that groups of similar objects have lower entropy than those of dissimilar ones. Thus, the COOLCAT algorithm [3] proposes a scheme where data objects are processed incrementally, and a suitable cluster is chosen for each tuple such that at each step, the entropy of the resulting clustering is minimized. The scaLable InforMation BOttleneck (LIMBO) algorithm [2] also exploits a notion of entropy to catch the similarity between objects and defines a clustering procedure that minimizes the information loss. The algorithm builds a *Distributional Cluster Features* (DCF) tree to summarize the data in k clusters, where each node contains statistics on a subset of tuples. Then, given a set of k clusters and their corresponding DCFs, a scan over the data set is performed to assign each tuple to the cluster exhibiting the closest DCF. The generation of the DCF tree is parametric to a user-defined branching factor and an upper bound on the distance between a leaf and a tuple. Furthermore, its time complexity is claimed to be $O(nm + m^2)$.

Li and Ma [38] propose an iterative procedure that is aimed at finding the optimal data partition that minimizes an entropy-based criterion. Initially, all tuples reside within a single cluster. Then, a Monte Carlo process is exploited to randomly pick a tuple and assign it to another cluster as a trial step aimed at decreasing the entropy criterion. Updates are retained whenever entropy diminishes. The overall process is iterated until there are no more changes in cluster assignments. Interestingly, the entropy-based criterion proposed here can be derived in the formal framework of probabilistic clustering models. Indeed, appropriate probabilistic models, namely, multinomial [7] and multivariate Bernoulli [8], have been proposed and shown to be effective. It is worth noticing that the classical Expectation-Maximization framework [29], equipped with any of these models, reveals to be particularly suitable for dealing with transactional data [30], [35], being scalable both in n and in m. Still, the correct estimation of an appropriate number of mixtures, as well as a proper initialization of all the model parameters, is problematic here.

The problem of estimating the proper number of clusters in the data has been widely studied in the literature. Many existing methods focus on the computation of costly statistics based on the within-cluster dispersion [17] or on cross-validation procedures for selecting the best model [12], [36]. The latter requires an extra computational cost due to a repeated estimation and evaluation of a predefined number of models. More efficient schemes have been devised in [34], [15], [37]. Starting from an initial partition containing a single cluster, the approaches iteratively apply the K-Means algorithm (with $k = 2$) to each cluster so far discovered. The decision on whether to switch the original cluster with the newly generated subclusters is based on a quality criterion, for example, the Bayesian Information Criterion [34], which mediates between the likelihood of the data and the model complexity, or the improvement in the rate of distortion (the variance in the data) of the subclusters with respect to the original cluster [37]. The exploitation of the K-Means scheme makes the algorithm specific to low-dimensional numerical data, and proper tuning to high-dimensional categorical data is problematic.

Also, automatic divisive approaches that adopt the classical top-down induction of decision trees have been proposed [4], [6], [27]. In these approaches, the attribute with the largest gain in the quality function is selected, and a split is performed according to the possible values of such an attribute. The approaches differ in the quality criterion adopted: reduction in entropy [4], [27] or distance among the prototypes of the resulting clusters [6]. Compared with [4], [6], the approach in [27] establishes a different connection to supervised learning. The authors introduce a binary target attribute and label each tuple in the data set with the "yes" label. Then, new artificial (*nonexisting*) data points, exhibiting the "no" label, are uniformly distributed within the data set. A decision tree learning algorithm can hence be applied to the augmented data set, and the resulting decision boundaries allow them to separate dense regions of "yes" data points. The authors concentrate on how they can avoid the physical generation of artificial data points and an overly excessive number of splits.

All of these approaches share some of the drawbacks previously described (for example, the incapability of discovering clusters with overlapping features). As an example, the approach in [27] may separate the same cluster (especially those of irregular shape) into multiple pieces, in an attempt of drawing decision boundaries. Moreover, they poorly scale on high-dimensional data: since each step of the clustering process requires each attribute to be evaluated for splitting, a large number of attributes would require a prohibitive number of evaluations.

3 The AT-DC Algorithm

We begin by fixing a proper notation to be used throughout the paper. Let us consider a set $\mathcal{M} = \{a_1, \ldots, a_m\}$ of Boolean attributes and a data set $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ of tuples defined on $\mathcal{M}$. In the current literature, $a \in \mathcal{M}$ is usually denoted as an *item*, and a tuple $\mathbf{x} \in \mathcal{D}$ as a *transaction*. Usually, $\mathbf{x}$ is represented in a more compact form as a proper subset of $\mathcal{M}$, with the meaning that all the items explicitly represented in $\mathbf{x}$ take value *true*, and the others take value *false*. Data sets composed of transactions are usually denoted as *Transactional data*, which is a special case of *high-dimensional categorical data* (that is, data which adhere to a schema where there are several not necessarily

Boolean attributes). Notice that each data set whose attributes are categorical can be represented as transactional data (by explicitly representing each possible attribute value as a Boolean attribute in $\mathcal{M}$). Hence, in the following, we shall refer mainly to transactional data.

A *cluster* is any set $\mathcal{S} \subseteq \mathcal{D}$. We denote by $n_{\mathcal{S}}$ the size of $\mathcal{S}$, and by $m_{\mathcal{S}}$ the size of $\mathcal{M}_{\mathcal{S}} = \{a | a \in \mathbf{x}, \mathbf{x} \in \mathcal{S}\}$. A partitioning problem consists of dividing the original collection of data $\mathcal{D}$ into a set $\mathcal{P} = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$ of nonempty clusters $\mathcal{C}_j$ such that each cluster contains a homogeneous subset of transactions. The notion of homogeneity found several different formalizations in the literature. In this paper, we relate the notion of homogeneity with the degree of overlap within transactions: Clusters where transactions share several items exhibit higher homogeneity than other subsets where transactions share few items. As a consequence, a cluster of transactional data is a set of tuples where certain items occur with higher frequency than elsewhere. Notice that highly frequent items implicitly define dense regions (subspaces) where clusters occur. Hence, clustering transactional data can be stated as the problem of detecting proper subspaces, which indeed define clusters.

Our approach to clustering starts from the analysis of the analogies between a clustering problem and a classification problem. In both cases, a model is evaluated on a given data set, and the evaluation is positive when the application of the model locates fragments of the data exhibiting high homogeneity. A simple rather intuitive and parameter-free approach to classification is based on decision tree learning, which is often implemented through top-down divide-and-conquer strategies. Here, starting from an initial root node (representing the whole data set), iteratively, each data set within a node is split into two or more subsets, which define new subnodes of the original node. The criteria upon which a data set is split (and, consequently, a node is expanded) is based on a quality criterion: choosing the best "discriminating" attribute (that is, the attribute producing partitions with the highest homogeneity) and partitioning the data set on the basis of such attribute. The concept of homogeneity has found several different explanations (for example, in terms of entropy or variance) and, in general, is related to the different frequencies of the possible labels of a target class.

The key idea of our approach is to develop a clustering procedure, which resembles the general schema of a top-down decision tree learning algorithm. We start from an initial partition containing a single cluster (representing the whole data set) and then iteratively try to split a cluster within the partition into two subclusters. If the subclusters guarantee a higher homogeneity in the partition than the original cluster, the latter is removed, and the outcome of splitting is added to the partition. The approach is based on the capability of splitting the clusters on the basis of their homogeneity. We assume that a function $Quality(\mathcal{C})$ measures the degree of homogeneity of a cluster $\mathcal{C}$. In practice, clusters with high intrahomogeneity exhibit high values of $Quality$.

The general schema of the AT-DC algorithm, implementing the envisaged procedure, is specified in Fig. 1. The algorithm starts with a partition containing a single cluster corresponding to the whole data set (line 1). The core of the algorithm is the body of the loop between lines 2 and 15. Within the loop, an attempt to generate a new cluster is performed by 1) choosing a candidate node (corresponding to a cluster with low quality) to split (line 4), 2) splitting the candidate cluster into two subclusters (line 5), and 3) evaluating whether the splitting allows a new partition that exhibits better quality than the original partition (lines 6–13). If this is the case, the loop can be stopped (line 10), and the partition is updated by replacing the candidate cluster with the new subclusters (line 8). Otherwise, the subclusters are discarded, and a new candidate cluster is considered for splitting.

```
GENERATE-CLUSTERS(𝒟)
  Input: A set 𝒟 = {x_1, ..., x_N} of transactions;
  Output: A partition 𝒫 = {𝒞_1, ..., 𝒞_k} of clusters;
1:  Let initially 𝒫 = {𝒟};
2:  repeat
3:    Generate a new cluster 𝒞 initially empty;
4:    for each cluster 𝒞_i ∈ 𝒫 do
5:      PARTITION-CLUSTER(𝒞_i, 𝒞);
6:      𝒫' ← 𝒫 ∪ {𝒞};
7:      if Quality(𝒫) < Quality(𝒫') then
8:        𝒫 ← 𝒫';
9:        STABILIZE-CLUSTERS(𝒫);
10:       break
11:     else
12:       Restore all x_j ∈ 𝒞 into 𝒞_i;
13:     end if
14:   end for
15: until no further cluster 𝒞 can be generated
```

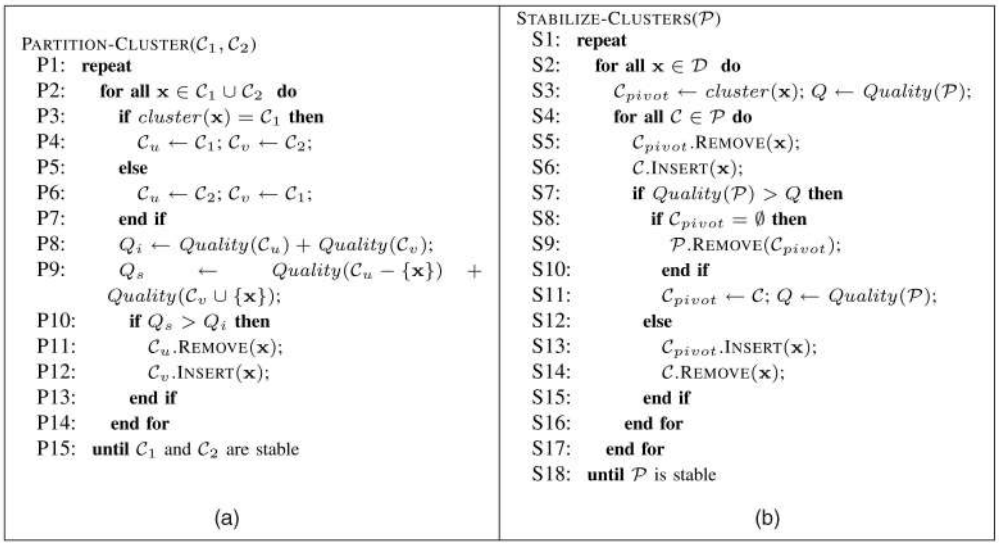
Fig. 1. The AT-DC scheme.

The generation of a new cluster triggers the call to STABILIZE-CLUSTERS in line 9, which aims at further improving the overall quality by attempting relocations among the clusters. Also, clusters at line 4 are considered in increasing order of quality. This guarantees that the effects of splitting are evaluated first on clusters with lower quality. Indeed, if a cluster exhibits a lower degree of homogeneity, it is more eligible for producing an improvement in the overall quality, provided that it is properly split.

3.1 Splitting a Cluster

It is clear from the above discussion that the heart of the proposed algorithm is a splitting procedure, which guarantees a significant improvement in the quality of the partition. Traditional decision tree learning algorithms choose the attribute that best discriminates among the classes by producing a partition for each possible value of the attribute. Candidate partitions are obtained *for free*, since there is a fixed number of candidate partitions to examine, and each partition is associated with a value of an attribute. Moreover, the overhead due to the identification of an appropriate partition is, in general, linear in the size of the data.

In principle, the same criteria could be applied here by choosing the attribute that guarantees the highest improvement in the quality of the partition. Indeed, the approaches in [6], [4] implement similar strategies. We found such strategies to be unsuitable to large values of m. In sparse data sets, an item a is likely to appear in few transactions. Whenever a co-occurs with other items, a splitting based on



PARTITION-CLUSTER($\mathcal{C}_1, \mathcal{C}_2$)
P1: **repeat**
P2: **for all** $\mathbf{x} \in \mathcal{C}_1 \cup \mathcal{C}_2$ **do**
P3: **if** $cluster(\mathbf{x}) = \mathcal{C}_1$ **then**
P4: $\mathcal{C}_u \leftarrow \mathcal{C}_1$; $\mathcal{C}_v \leftarrow \mathcal{C}_2$;
P5: **else**
P6: $\mathcal{C}_u \leftarrow \mathcal{C}_2$; $\mathcal{C}_v \leftarrow \mathcal{C}_1$;
P7: **end if**
P8: $Q_i \leftarrow Quality(\mathcal{C}_u) + Quality(\mathcal{C}_v)$;
P9: $Q_s \leftarrow Quality(\mathcal{C}_u - \{\mathbf{x}\}) + Quality(\mathcal{C}_v \cup \{\mathbf{x}\})$;
P10: **if** $Q_s > Q_i$ **then**
P11: $\mathcal{C}_u$.REMOVE($\mathbf{x}$);
P12: $\mathcal{C}_v$.INSERT($\mathbf{x}$);
P13: **end if**
P14: **end for**
P15: **until** $\mathcal{C}_1$ and $\mathcal{C}_2$ are stable

(a)

STABILIZE-CLUSTERS($\mathcal{P}$)
S1: **repeat**
S2: **for all** $\mathbf{x} \in \mathcal{D}$ **do**
S3: $\mathcal{C}_{pivot} \leftarrow cluster(\mathbf{x})$; $Q \leftarrow Quality(\mathcal{P})$;
S4: **for all** $\mathcal{C} \in \mathcal{P}$ **do**
S5: $\mathcal{C}_{pivot}$.REMOVE($\mathbf{x}$);
S6: $\mathcal{C}$.INSERT($\mathbf{x}$);
S7: **if** $Quality(\mathcal{P}) > Q$ **then**
S8: **if** $\mathcal{C}_{pivot} = \emptyset$ **then**
S9: $\mathcal{P}$.REMOVE($\mathcal{C}_{pivot}$);
S10: **end if**
S11: $\mathcal{C}_{pivot} \leftarrow \mathcal{C}$; $Q \leftarrow Quality(\mathcal{P})$;
S12: **else**
S13: $\mathcal{C}_{pivot}$.INSERT($\mathbf{x}$);
S14: $\mathcal{C}$.REMOVE($\mathbf{x}$);
S15: **end if**
S16: **end for**
S17: **end for**
S18: **until** $\mathcal{P}$ is stable

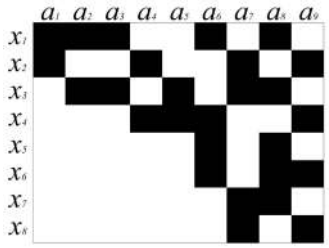
(b)

Fig. 2. Local/Global element relocation.

a would produce low-quality partitions. Consider, for example, the following transactions:

$$\begin{array}{ll} \mathbf{x}_1 = \{a_1, a_2, a_3, a_6, a_8\}, & \mathbf{x}_2 = \{a_1, a_4, a_7, a_9\}, \\ \mathbf{x}_3 = \{a_2, a_3, a_5, a_7, a_8\}, & \mathbf{x}_4 = \{a_4, a_5, a_6, a_9\}, \\ \mathbf{x}_5 = \{a_6, a_8\}, & \mathbf{x}_6 = \{a_6, a_8, a_9\}, \\ \mathbf{x}_7 = \{a_7, a_8\}, & \mathbf{x}_8 = \{a_7, a_9\}. \end{array}$$

There is a neat separation between transactions in $S_1 = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$ and $S_2 = \{\mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7, \mathbf{x}_8\}$. This can be better noticed in the following bitmapped representation:



Indeed, none of the transactions in S_1 contains items $a_1 - a_5$, which, on the contrary, appear within the transactions in S_2. Also, transactions in each group exhibit a significant overlap. However, a split based on any item would separate the transactions from S_1 and mix them with the ones in S_2.

The point is that, as described in [14], [41], clusters of categorical data are often characterized by frequent and partially overlapping regions, where a region represents a subset of items. In the above example, each item represents a region, and all regions overlap in the transactions containing them. Hence, choosing partitions on the basis of a single attribute only would likely split overlapping regions, thus lowering the quality of the resulting partitions.

To avoid the above drawbacks, we resort here to a greedy approach, which starts from an initial partition and then progressively moves to *neighbor* partitions as soon as there is a convenience in doing so. Here, two neighbor partitions P and P' differ over the assignment of a single element $\mathbf{x}$: precisely, $\mathbf{x}$ belongs to $\mathcal{C}_i$ in P and to $\mathcal{C}_j \neq \mathcal{C}_i$ in P'.

The PARTITION-CLUSTER algorithm in Fig. 2a implements such a schema. The algorithm is characterized by an iterative analysis of the elements involved in the splitting. The algorithm iteratively evaluates, for each element $\mathbf{x} \in \mathcal{C}_1 \cup \mathcal{C}_2$, whether a membership reassignment improves the degree of homogeneity of the two clusters. Homogeneity is evaluated on a local basis by combining the contribution of each of the two clusters involved. Lines P8 and P9 compute the contribution of $\mathbf{x}$ to the local quality in two cases: either in the case that $\mathbf{x}$ is maintained in its original cluster of membership (represented by $\mathcal{C}_u$) or in the case that $\mathbf{x}$ is moved to the other cluster (represented by $\mathcal{C}_v$). If moving $\mathbf{x}$ causes an improvement in the local quality, then the swap is accepted (lines P10–P13).

Lines P2–P14 in the algorithm are nested into a main loop: elements are iteratively checked for swapping until a convergence is met. In practice, the algorithm continues reassigning elements from one cluster to another until a convenience in swapping is found. In a sense, the scheme of the algorithm is a generalization of the *K-Means* scheme (for $k = 2$). Indeed, the quality of a cluster can be defined in terms of neighborhood of its elements to a cluster prototype. Thus, relocation is accomplished only if an element is nearer the opposite cluster's prototype.

It is easy to see that the algorithm converges after a finite number of steps.

Theorem 3.1. *The algorithm in Fig. 2a terminates for any initial values of $\mathcal{C}_1$ and $\mathcal{C}_2$.*

Proof. Since two partitions can be compared according to their quality criterion, the space of all possible partitions is a lattice under such a comparison. At each iteration of the loop described by lines P2–P14, the algorithm traverses at most n neighbors (where n is the size of $\mathcal{C}_1 \cup \mathcal{C}_2$), updating the current structure to neighbors exhibiting higher quality. Thus, the algorithm traverses the lattice of all partitions upward. Termination is inferred by observing that such a lattice is finite. □

Some further aspects deserve to be discussed. The first is that even though convergence is guaranteed, the computational complexity could, in principle, be high: The number of possible configurations is exponential, and there is no apparent guarantee that the algorithm will not consider all of them. However, this is very unlikely to happen. Intuitively, each cluster acts as a gravity center by attracting elements that contemporarily improve its quality and reduce the loss in the opposite cluster. Under this perspective, it is unlikely that once relocated, an element is further considered for swapping in a subsequent step. If a single element only were relocated for each iteration, the cost of the main loop would be quadratic in the size of $\mathcal{C}_1 \cup \mathcal{C}_2$. Nevertheless, single relocations seldom occur and, consequently, PARTITION-CLUSTER exhibits a practically faster rate of convergence. This is confirmed in Section 4.

Finally, there is no guarantee that the split computed by the PARTITION-CLUSTER procedure is a global maximum. Indeed, the way that the original cluster $\mathcal{C}_1$ is split still depends on 1) the initial point, which is moved into $\mathcal{C}_2$ first, and 2) the order in which the elements are considered for relocation. As to the former point, the first optimization can be achieved by forcing a refined initialization of $\mathcal{C}_2$ by choosing the element $\mathbf{x}$ whose removal from $\mathcal{C}_1$ best improves its quality.

However, the splitting process can still be sensitive to the order upon which elements are considered: In the first stage, it could be not convenient to reassign the generic $\mathbf{x}_i$ from $\mathcal{C}_1$ to $\mathcal{C}_2$, whereas a convenience in performing the swap can be found after the relocation of some other element $\mathbf{x}_j$. The main loop partly smooths this effect by repeatedly relocating objects until convergence is met. However, better PARTITION-CLUSTER can be made strongly insensitive to the order with which cluster elements are considered. The basic idea is discussed next. In general, the elements that mostly influence the locality effect are either outlier transactions (that is, those containing mainly items, whose frequency within the cluster is rather low) or common transactions (which, dually, contain very frequent items). In the first case, $\mathcal{C}_2$ is unable to attract further transactions, whereas in the second case, $\mathcal{C}_2$ is likely to attract most of the transactions (and, consequently, $\mathcal{C}_1$ will contain outliers).

Thus, the key idea is to rank and sort the cluster elements before line P1, which is on the basis of their splitting effectiveness. To this purpose, each transaction $\mathbf{x}$ belonging to cluster $\mathcal{C}$ can be associated with a weight $w(\mathbf{x})$, which indicates its splitting effectiveness. Ideally, $\mathbf{x}$ is eligible for splitting $\mathcal{C}$ if its items allow us to divide $\mathcal{C}$ into two homogeneous subclusters. In this respect, the Gini index is a natural way to quantify the splitting effectiveness $G(a)$ of the individual attribute value $a \in \mathbf{x}$. Precisely, $G(a) = 1 - \Pr(a|\mathcal{C})^2 - (1 - \Pr(a|\mathcal{C}))^2$, where $\Pr(a|\mathcal{C})$ denotes the probability of a within $\mathcal{C}$. Notice that $G(a)$ is close to its maximum whenever a is present in about half of the transactions of $\mathcal{C}$ and reaches its minimum whenever a is unfrequent or common within $\mathcal{C}$. The overall splitting effectiveness of $\mathbf{x}$ can be defined by averaging the splitting effectiveness of its constituting items $w(\mathbf{x}) = avg_{a \in \mathbf{x}}(G(a))$. Once ranked, the elements $\mathbf{x} \in \mathcal{C}$ can be considered in descending order of their splitting effectiveness at line P2. This guarantees that $\mathcal{C}_2$ is initialized with elements, which do not represent outliers and still are likely to be removed from $\mathcal{C}_1$. In most cases, this removes the dependency on the initial input order of the data.

Notice that as with decision tree learning, AT-DC exhibits a preference bias, which is encoded within the notion of homogeneity and can be viewed as the preference for compact clustering trees. Indeed, due to the splitting effectiveness heuristic, homogeneity is enforced by the effects of the Gini index. At each split, this tends to isolate clusters of transactions with mostly frequent attribute values, from which the compactness of the overall clustering tree follows.

As the final remark, the aforementioned heuristic based on transaction splitting effectiveness determines a negligible increase in the computational cost of the PARTITION-CLUSTER procedure. Indeed, since element sorting is done preliminarily to the execution of the main loop, the overall cost of PARTITION-CLUSTER is raised by a quasilinear contribution.

3.2 Stabilizing Clusters

PARTITION-CLUSTER attempts to improve the local quality of a cluster. In contrast, the STABILIZE-CLUSTERS procedure tries to increase partition quality by finding, for each element, the most suitable cluster among the ones available in the partition. Fig. 2b shows the pseudocode of the procedure. Again, the core of the algorithm is a main loop whose body (lines S2–S17) examines all the available elements. For each element $\mathbf{x}$, a pivot cluster is tracked, which is the cluster currently including $\mathbf{x}$. Initially, the pivot cluster is the cluster that contains $\mathbf{x}$. Then, the available clusters are iteratively evaluated. The insertion of $\mathbf{x}$ in the current cluster (and its consequent removal from $\mathcal{C}_{pivot}$) is attempted (lines S5–S6), and the updated quality is compared with the original quality. If an improvement is obtained, then the swap is accepted (line S11). The new pivot cluster is the one now containing $\mathbf{x}$, and if the removal of $\mathbf{x}$ causes emptying the old pivot cluster, the latter is removed from the partition $\mathcal{P}$. Conversely, if there is no improvement in quality, $\mathbf{x}$ is restored into its pivot cluster, and a new cluster is examined.

The main loop is iterated until a stability condition for cluster membership is met. Convergence of STABILIZE-CLUSTERS can be evicted by an argument that is similar to the one used for PARTITION-CLUSTER. Again, the rate of convergence of the procedure is very high: since the procedure is called only after a new cluster is generated, only local rearrangements are needed. Typically, they involve elements that are more attractive for the newly generated clusters and, in the previous steps of the GENERATE-CLUSTERS procedure, were assigned to different clusters. Experimentally, we found that convergence is usually reached in not more than two steps. This explains why the result of the PARTITION-CLUSTER procedure in the GENERATE-CLUSTERS algorithm is typically a new cluster $\mathcal{C}$ with a quite high degree of homogeneity. That is, at each invocation, the PARTITION-CLUSTER procedure identifies a kernel of elements, which form a natural cluster and seldom need a rearrangement into new clusters.

The only situation that, in principle, could cause massive rearrangements is the fortuitous splitting of a natural cluster into two or more subclusters. However, in such a case, an adequate quality measure should foster the merging of the two clusters and the removal of the redundant one (envisaged in lines S8 and S9 of the scheme) by penalizing the overall quality.

3.3 Cluster and Partition Qualities

AT-DC exploits two different quality measures, namely, one for the local homogeneity within a cluster and another for the global homogeneity of the partition. These measures are employed with opposing objectives. Indeed, as shown in Fig. 1, we can notice that partition quality is used for establishing whether the insertion of a new cluster is really convenient: In a sense, it is aimed at maintaining compactness. Conversely, cluster quality in procedure PARTITION-CLUSTER is aimed at the best localized splitting and, hence, at a good separation.

In principle, cluster quality is identified with a high degree of intracluster homogeneity, as well as with its separation from other clusters. As stated in [11], there is a strict relation between intracluster homogeneity and the probability $\Pr(a_i|\mathcal{C}_k)$ that item a_i appears in a transaction belonging to $\mathcal{C}_k$: The higher the value of such a probability, the higher the number of transactions in $\mathcal{C}_k$ sharing a_i. Moreover, there is a strict correlation between intercluster separation and $\Pr(\mathbf{x} \in \mathcal{C}_k | a_i \in \mathbf{x})$: A high value of such a probability for $\mathcal{C}_k$, corresponds to low probability values for clusters other than $\mathcal{C}_k$ (and, hence, to few transactions containing a_i in such clusters). Thus, cluster homogeneity and separation can be effectively computed without resorting to costly similarity measures but by simply relating it to the commonality of items within the transactions that it contains. In particular, cluster quality can be expressed to be proportional to the combination of the above probabilities $\sum_{a \in \mathcal{M}_\mathcal{C}} \Pr(a|\mathcal{C}) \Pr(\mathcal{C}|a) \Pr(a)$. The last term is used for weighting the importance of item a in the summation: Essentially, high values from low-frequency items are less relevant than those from high-frequency values. By the Bayes theorem, the above formula can be expressed as $\Pr(\mathcal{C}) \sum_{a \in \mathcal{M}_\mathcal{C}} \Pr(a|\mathcal{C})^2$. Terms $\Pr(a|\mathcal{C})^2$ (that is, the relative strength of a within $\mathcal{C}$) and $\Pr(\mathcal{C})$ (that is, the relative strength of $\mathcal{C}$) work in contraposition: Singleton clusters exhibit strong items in a sparse region, whereas highly populated clusters exhibit weaker items in a dense region. In practice, it is more convenient to compute, for each item, the gain in strength with respect to the whole data set, that is

$$Quality(\mathcal{C}_k) = \Pr(\mathcal{C}_k) \sum_{a \in \mathcal{M}_{\mathcal{C}_k}} \Big[\Pr(a|\mathcal{C}_k)^2 - \Pr(a|\mathcal{D})^2 \Big].$$

The above formula also finds an interpretation in terms of subspace clustering. In fact, items exhibiting higher occurrence frequency compared to the occurrence frequency in the original data set define a subset of relevant features, as opposed to low-occurrence items, which are indeed irrelevant to the purpose of clustering. Thus, clusters exhibit high quality whenever a subspace of relevant items occurs, whose frequency is significantly higher than in the original data set.

Despite their similarity, there is a subtle and substantial difference between the above definition of quality and the definition provided by Fisher in [11]. There, each possible attribute value contributes to the quality. In a sense, this corresponds to give equal importance to both the presence and the absence of an item within a transaction. In other words, a faithful application of the concept of *category utility* would result in the redefinition of the summation in the above formula as

$$\sum_{a \in \mathcal{M}} \Big[\Big(\Pr(a|\mathcal{C}_k)^2 - \Pr(a|\mathcal{D})^2 \Big) + \Big(\Pr(\neg a|\mathcal{C}_k)^2 - \Pr(\neg a|\mathcal{D})^2 \Big) \Big].$$

As a drawback, in a high-dimensional setting, the gain in the first term would be counterbalanced by a loss in the second term, thus invalidating the approach. In other words, an item a with a probability of appearing in $\mathcal{C}_k$, which is significantly higher than the probability of appearing in $\mathcal{D}$, would give no contribution to the cluster quality.

In Fig. 1, the quality of a partition is meant to measure both the homogeneity of clusters and their compactness. Viewed in this respect, partition quality can be defined as the weighted sum of the qualities of the available clusters:

$$Quality(\mathcal{P}) = \sum_{\mathcal{C} \in \mathcal{P}} \Pr(\mathcal{C}) Quality(\mathcal{C}).$$

The formula finds an interpretation in terms of the average increase in quality obtained by partitioning the data set. Notice that the component $Quality(\mathcal{C})$ is already proportional to the contribution $\Pr(\mathcal{C})$. As a result, in the overall partition quality, the contribution of each cluster is weighted by $\Pr(\mathcal{C})^2$. This weighting has a major effect in the GENERATE-CLUSTERS procedure: Splitting in extremely small clusters is penalized. Indeed, the generated clusters are added to the partition only if their contribution is really worth. Notice the different roles of the contribution $\Pr(\mathcal{C})$ in the two quality measures: A strong penalization on singleton clusters would not allow a proper splitting in the PARTITION-CLUSTER procedure. In particular, splitting would suffer from the bottleneck of the initial reassignment, since, in principle, the possible loss in cluster $\mathcal{C}_1$ would not be counterbalanced by a proper gain in $\mathcal{C}_2$. On the contrary, the (stable) result of the PARTITION-CLUSTER procedure has to be accepted only if it yields a significant change in the average cluster quality. The same interpretation justifies the usage of partition quality in the STABILIZE-CLUSTERS procedure. There, massive rearrangements should be discouraged, since the main objective of the procedure is to further purge existing clusters on the basis of the newly discovered splitting.

4 Evaluation

Hereafter, we analyze the behavior of the AT-DC proposed in the previous section. The analysis is performed with two main objectives:

- *The assessment of result quality*. Since clustering algorithms define structures that are not known a priori, irrespective of the clustering methods, the final partition of data requires some kind of evaluation [22]. The evaluation is useful for measuring the adequacy of the discovered structure so that it can be interpreted objectively. The adequacy of a clustering structure refers to the extent in which the clustering structure provides true information about the data: A clustering structure is *valid* if it fits the data set, that is, if the discovered clusters correspond to the actual homogeneous groups in the data set.
- *An extensive comparative analysis of its performance*. Since the proposed algorithm is specifically designed to deal with a large data set of high dimensionality, it is important to measure its performance on the basis of such parameters. Also, notice that since the computational complexity cannot be analytically devised, an experimental performance analysis is critical in the approach. To this purpose, it is important to provide an efficient implementation of the algorithm, which works effectively, even in "extreme" situations.

Experiments are conducted on both real and synthesized data, and their results are cross validated across different orders of the data. Precisely, several real-life data sets are employed in Section 4.3 to evaluate the effectiveness of AT-DC. In particular, the algorithm is tested on the UCI Machine Learning Repository for the purpose of enabling an easier comparison with different approaches from the current literature.

Also, several synthesized data sets are exploited in Section 4.4 to investigate scalability and robustness in critical applicative settings, and a selection of real-life and synthesized data sets is leveraged in Section 4.5 to compare AT-DC against some major algorithms for categorical clustering.

4.1 Implementation Details

The implementation of the algorithm needs to face some crucial issues, which may severely affect its efficiency. First, it must be noticed that the most frequent operation invoked by the algorithm is the computation, for a given cluster $\mathcal{C}$, of $Quality(\mathcal{C})$. Hence, it is crucial to maintain the cost of such an operation as low as possible. A naive implementation requires $O(m)$ updates, which is clearly not viable. However, the local quality can be estimated as

$$\begin{aligned} Quality(\mathcal{C}) &= \Pr(\mathcal{C}) \sum_{a\in\mathbf{x},\mathbf{x}\in\mathcal{C}} \left[\Pr(a|\mathcal{C})^2 - \Pr(a)^2\right] \\ &\approx \frac{n}{N} \sum_{a\in\mathbf{x},\mathbf{x}\in\mathcal{C}} \left[\left(\frac{n_a}{n}\right)^2 - \left(\frac{N_a}{N}\right)^2\right] \\ &\approx \frac{1}{n_{\mathcal{C}} n} \sum_{a\in\mathbf{x},\mathbf{x}\in\mathcal{C}} n_a^2 - \frac{n_{\mathcal{C}}}{n^3} \sum_{a\in\mathbf{x},\mathbf{x}\in\mathcal{C}} N_a^2, \end{aligned} \quad (1)$$

where n_a and N_a represent the frequencies of a in $\mathcal{C}$ and $\mathcal{D}$, respectively. The components $\sum n_a^2$ and $\sum N_a^2$ can be incrementally maintained via the adoption of hash tables. Hence, the value of $Quality(\mathcal{C})$ can be updated as soon as a new transaction is added to $\mathcal{C}$ at a practically constant cost.

A further issue consists of the effective management of large data sets, which do not fit in the main memory. Since the execution of the algorithm causes a large number of swaps between clusters, attention must be kept in order to avoid disk thrashing. The potential problems are lines P9 and S5–S6 in the two auxiliary methods. Notice that in such cases, there is no effective need to actually perform such updates, since we only need to compute the change in quality (which is hence updateable in constant time, as shown above).

Rather, a more subtle problem arises when the hash tables used for collecting the frequencies of items in the above formula do not fit in the main memory. Since each transaction $\mathbf{x}$ evaluated for swapping requires updates to the frequencies, the risk is that each iteration, both in PARTITION-CLUSTER and in STABILIZE-CLUSTERS, costs $O(N)$ I/Os.

In practice, data sets with a dimensionality higher than 2^{25} are quite unusual. The latter is the maximum size allowance for a hash table requiring 64 bytes for each entry, which fits in the main memory of today's workstations. Further, notice that in such data sets, thrashing is produced by a huge number of low-frequency items. Notwithstanding, we can still process such data sets by approximating the terms in (2), without the need to collect and maintain all the frequencies. Indeed, each term in the formula can be estimated and computed incrementally by resorting to mean and variance of item frequencies. For example, $1/n \sum n_a^2 \approx avg(n_a) + var(n_a)$, where $avg(n_a)$ and $var(n_a)$ are, respectively, the sample mean and variance of item frequency. Now, it is easy to see that $avg_C(n_a) = 1/n_C \sum_i |\mathbf{x}_i|$. Let $\delta(\mathbf{x}, a)$ be the function returning 1 if $a \in \mathbf{x}$, and 0 otherwise. Then,

$$\begin{aligned} avg(n_a) &= 1/n_C \sum_a \sum_{\mathbf{x}\in C} \delta(\mathbf{x}, a) = 1/n_C \sum_{\mathbf{x}\in C} \sum_a \delta(\mathbf{x}, a) \\ &= 1/N \sum_{\mathbf{x}\in C} |\mathbf{x}|. \end{aligned}$$

Hence, a suitable approximation of the term $1/n \sum n_a^2$ can be obtained by estimating the variance $var(n_a)$ on a representative sample of the available items that fits in the main memory.

4.2 Quality Measures

Three main methods for assessing clustering effectiveness are described in [22], [25], namely,

- *External criteria*. When clustering results are evaluated according to a prespecified structure, which corresponds to a meaningful explanation of the data at hand.
- *Internal criteria*. When clustering results are evaluated in terms of the quantities that are computable from the available data.
- *Relative criteria*. When evaluation takes place in comparison with other clustering schemes.

It is worth recalling here that the adoption of external criteria helps in understanding clustering results and, hence, the adequacy of a clustering algorithm. Indeed, a predefined structure can be interpreted as the explanation

of the corresponding data by means of some hypotheses. As a consequence, the correspondence of a cluster to one of such predefined structures implies the interpretation of such a cluster according to the corresponding hypothesis. For these reasons, we resort to external criteria in order to evaluate the partitioning quality. More specifically, we investigate the behavior of the AT-DC algorithm on collections of data with class labels and analyze the correspondence between the discovered and hypothesized structures. Notice that we exploit standard benchmark data sets with inherent structures of strong regularities, which would allow a classifier to accurately distinguish among the corresponding cases. In particular, class labels serve as ground truth for natural data classes. Therefore, by matching the discovered partitions against the actual data classes, we measure the effectiveness of both AT-DC and other state-of-the-art algorithms in discovering natural clusters.

In order to evaluate such a correspondence, we exploited the *indicator functions* I_D and I_H [25], where $I_D(\mathbf{x}_i, \mathbf{x}_j) = 1$ if both $\mathbf{x}_i$ and $\mathbf{x}_j$ are assigned by the algorithm to the same cluster, and 0 otherwise. Analogously, $I_H(\mathbf{x}_i, \mathbf{x}_j) = 1$ if both $\mathbf{x}_i$ and $\mathbf{x}_j$ are in the same (true) class. We employed the above indicators to build a *contingency table* describing the number a of pairs that are in the same class and in the same cluster, the number b of pairs that are in the same cluster but in different classes, the number c of pairs that are in the same class but in different clusters, and the number d of pairs that are in different classes and in different clusters. Then, the following indices can be defined on the contingency table:

- *Rand*. This is defined as $(a+d)/(a+b+c+d)$ and represents the compactness and separability of the discovered structure related to the hypothesized structure.
- *Jaccard* and *Fowlkes*. These are defined as $a/(a+b+c)$ and $a/\sqrt{(a+b)(a+c)}$, respectively, and represent the compactness of the discovered structure.
- Γ. This is the normalized correlation between I_D and I_H.

In addition, for each clustering result, we computed a further contingency table m, where columns represent discovered clusters, and rows represent true classes. The term m_{ij} corresponds to the number of transactions in $\mathcal{D}$ that were associated with cluster $\mathcal{C}_j$ and actually belong to the ideal class C_i.

The matrix provides an immediate visual description of the degree of agreement between the AT-DC and the ideal partitions. Intuitively, each cluster $\mathcal{C}_j$ corresponds to the class C_i that is best represented in $\mathcal{C}_j$ (that is, such that m_{ij} is maximal). A further measure can be devised, which reflects such a degree of correspondence. This is the *error rate* of the clustering scheme, that is, the number of instances that are misclassified in the contingency table:

$$\epsilon = \frac{\sum_j \sum_{i \neq h(j)} m_{ij}}{\sum_{i,j} m_{ij}},$$

where $h(j)$ is the index of the class C_i with maximal m_{ij}.

In general, misclassified items are produced by wrong allocations, that is, transactions whose actual class is not the predominant one within a cluster.[1] The evaluation of a clustering, according to an external criterion, is satisfactory when the number of wrong allocations is minimal. In particular, large values of the *Rand*, *Jaccard*, *Fowlkes*, and Γ indices represent high-quality clustering structures. Low values are instead preferable for ϵ, since it represents the error of the discovered structure in overlapping the hypothesized structure. Thus, a comparison of such values gives us an objective criterion for evaluating the results of the AT-DC algorithm.

4.3 Real-Life Data Sets

Several real-life data sets were evaluated. For each of these data sets, we measured the performance of AT-DC, with data ordering both enabled and disabled: In most cases, the results reported here were obtained by enabling the data ordering. A description of each data set employed for testing is provided next, together with an evaluation of the AT-DC performances.

4.3.1 SMART

The first data set that we analyze is the *SMART* collection from Cornell University.[2] This collection consists of 3,891 text documents organized into three main subcollections:

- *Medline*. This collection contains 1,033 abstracts from medical journals.
- *Cisi*. This collections contains 1,460 abstracts from information retrieval papers.
- *Cranfield*. This collection contains 1,398 abstracts from aeronautical system papers.

Through preprocessing, we obtained a series of data sets with increasing dimensionality, where a fixed dimensionality m was obtained by choosing the m most frequent terms. The corresponding data set was then obtained by representing each document as an item set with the selected terms. By applying the algorithm to such a data set, we obtained high-quality clusters. In particular, Fig. 3a details the results for the data set of dimensionality 14.571 (the results for different dimensionalities are summarized in Fig. 11). As we can see, the error rate is quite low (only 45 objects were misclassified). Moreover, the compactness and separability are quite good.

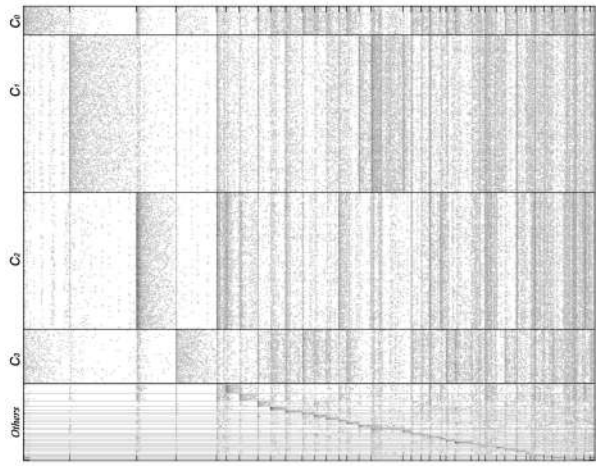
It is also interesting to analyze the consistency of clusters with respect to the items. The data set can be represented as a Boolean incidence matrix, where each row represents a transaction, each column an item, and a cell contains one if the item corresponding to the column belongs to the transaction corresponding to the row. Hence, a partition of the matrix in blocks, where each block represents a cluster, can give us a visual perception of the quality of the clustering result. Ideally, a good clustering would produce a block-triangular matrix, provided that a suitable sorting of the items is produced. In our representation, we sorted by

1. However, wrong allocations can also be due to outliers, that is, transactions sharing few (but not most) items with other transactions within a cluster. This is particularly likely to happen when the data is truly high dimensional.

2. ftp://ftp.cs.cornell.edu/pub/smart.

Cluster No.	*Medline*	*Cran*	*Cisi*
0	243	6	2
1	1	1	1351
2	0	1170	0
3	477	0	0
4	22	0	0
5	0	78	0
6-16	190	0	0
17-20	0	0	71
21-22	0	24	0
23	4	55	0
24	30	0	1
25	8	4	0
26	1	0	24
27	1	14	0
28	12	5	0
29	8	1	1
30	1	0	8
31	12	1	0
32	8	0	1
33	5	6	0
34	1	6	0
35	3	1	0
36	4	5	0
37	1	14	0
38	0	3	1
39	1	4	0

(a)



(b)

Fig. 3. Results for the *SMART* collection. (a) Contingency table. (b) Transaction/Item incidence matrix.

associating each item a to the cluster $\mathcal{C}$ exhibiting the highest $\Pr(a|\mathcal{C})$. The resulting incidence matrix for the *SMART* collection is shown in Fig. 3b.

Ticks in the figure represent cluster separators. In the figure, four large clusters are clearly visible and are represented by different items. Interestingly, the first and fourth clusters are quite similar, but the relevant items exhibit different frequencies. The high number of small clusters (indeed characterized by a small subset of items with high within-cluster probability) is due to the high variability of the subtopics within the same subcollection. The latter is a typical scenario occurring when data exhibit sparse high dimensionality: Each class is likely characterized by different subspaces, and several outliers can occur. Small clusters represent either subspaces or outliers.

Being explicitly designed for dealing with transactional data and due to the devised notion of cluster homogeneity, AT-DC is actually capable of identifying small groups of transactions consisting of items with low occurrence frequency within the corresponding clusters. Indeed, when a cluster is selected for splitting, its quality can be further improved by separating those transactions whose items do not frequently occur within them. This is testified by the fact that small clusters are usually detected at the last steps of the algorithm.

Cluster No.	Amphibian	Mammal	Bird	Reptile	Insect	Fish	Invertebrate
0	1	41	0	2	0	0	0
1	0	0	0	0	0	0	7
2	3	0	0	0	0	0	0
3	0	0	0	0	8	0	2
4	0	0	20	0	0	0	0
5	0	0	0	3	0	13	0
6	0	0	0	0	0	0	1

Fig. 4. Confusion matrix for zoo, with asymmetric representation.

4.3.2 UCI Data Sets

The behavior of the algorithm was also tested on several data sets from the UCI Machine Learning Repository. Generally, such data sets adhere to a fixed database schema, containing categorical and numerical attributes. The latter were disctretized by resorting to either *supervised* or *unsupervised* discretization. In our experiments, we represented the tuples within such data sets as transactions of fixed size, where each item is represented as a term "$attribute\text{-}name = attribute\text{-}value$." A description of each data set and the analysis of the AT-DC performances follow.

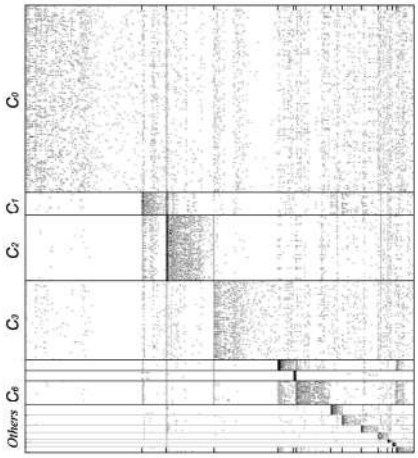
Zoo. The *zoo* data set is a simple database containing 101 records with 17 Boolean-valued attributes and describes the species of a set of animals on the basis of their attributes. Interestingly, the data set can be exploited as a toy example to understand the behavior of the algorithm. In particular it allows us to test our algorithm with two different representations of the data. Indeed, besides the traditional *symmetric* representation of each record as a set of Attribute/Value pairs, we can impose an *asymmetry* in the representation of the values. That is, we can choose to provide an interpretation of Boolean values in terms of transactions so that, for example, values $\{hair = true, feathers = false, eggs = false, milk = true\}$ can be represented simply as $\{hair, milk\}$ (that is, by discarding all the pairs such that the associated value is $false$). Fig. 7 shows the clustering results both for the symmetric and asymmetric representations. Both results exhibit good compactness and separability. In particular, it can be noticed in Fig. 4, that in the case of symmetric representation, the main sources of error are clusters 3 and 5: The first confuses among *insect* and *invertebrate*, whereas the second confuses *reptile* and *fish*. This is not surprising, since these two genders share several features.

Internet Ads. The *Advertising* data set contains 3,279 records and 1,554 Boolean attributes. In addition, three further attributes are "categorical" in nature (although they are numeric, several values occur frequently). To summarize, the total number of possible items is 2,832. This data set represents a set of possible advertisements on Internet pages. Each record represents a Web page, and the features encode phrases occurring in the URL, the image's URL and alt text, the anchor text, and words occurring near the anchor text. Each record is labeled either as "*ad*" or as "*noad*." The data set is quite unbalanced, since there are 2,821 "*noads*" and 458 "*ads*." AT-DC was tested on the *asymmetric* representations of such a data set. Notwithstanding, separability is quite good, as shown in Fig. 5a. In particular, notice how clusters 4 and 6 represent the minority class.

Fig. 5b reports the resulting incidence matrix. As it is clearly visible, the 14 clusters discovered by the algorithm are characterized by different subsets of items. The first, third, and fourth clusters are the largest in size, exhibiting the greatest high-density regions in the figure. The remaining

Cluster No.	No of ad.	No of nonad.
0	103	1263
1	3	163
2	27	450
3	52	525
4	78	2
5	0	75
6	170	3
7	0	74
8	2	75
9	0	53
10	0	50
11	0	23
12	0	30
13	24	19

(a)



(b)

Fig. 5. Ad, with asymmetric representation. (a) Contingency table. (b) Transaction/Item incidence matrix.

clusters are smaller in size, but each one includes a distinctive small subset of items. In particular, we point out that clusters 4 and 6 (representing the minority class) share some common items and still have distinguishing features: cluster 6 exhibits most of the features of cluster 4, plus some additional features.

Mushrooms. The *Mushroom* data set contains 8,124 records and 22 attributes. Each record describes one mushroom specimen in terms of 22 physical properties (for example, color, odor, size, and shape) and contains a label designating the specimen as either poisonous (3,916 records) or edible (4,208 records) too. All 22 attributes are categorical. We obtained the results detailed by the confusion matrix in Fig. 6a. As it can be seen, AT-DC automatically found eight clusters. From the confusion matrix, we can notice that only the clusters 1 and 4 contain misclassified objects, 360 and 192, respectively. However, cluster 1 (which is the main source of the error rate of AT-DC) has a net predominance of tuples labeled as Edible. Overall, the error rate is quite low because only 552 (out of 8,124) objects were misclassified. Finally, we can see a high homogeneity in clusters 3 and 5. Also, for such a data set, compactness and separability are quite good.

Congressional Votes. The *Congressional Votes* data set contains a set of US Congressional Voting Records. Each record corresponds to one Congressman's votes on 16 issues (for example, education spending, crime, and immigration).

Cluster No.	No of Edible	No of Poisonous
1	3536	360
2	288	0
3	0	1734
4	192	456
5	0	1296
6	0	64
7	192	0
8	0	6

(a)

Cluster No.	No of Democrats	No of Republicans
1	81	166
2	186	1

(b)

Cluster No.	No of Democrats	No of Republicans
1	202	25
2	46	0
3	7	64
4	12	78

(c)

Cluster No.	No of Democrats	No of Republicans
1	248	26
2	19	141

(d)

Fig. 6. Clustering results for Congressional Vote and Mushroom data. (a) Mushroom. (b) Vote, with symmetric representation. (c) Vote, with asymmetric representation. (d) Vote, with no data ordering.

All attributes are Boolean ("*Yes*" or "*No*" vote), and very few contain missing values. A classification label of "Republican" or "Democrat" is provided for each data record. The data set contains 435 records: 168 "Republican" and 267 "Democrat." We applied the algorithm to such a data set both in *asymmetric* and *symmetric* representations (where the latter is obtained by ignoring the "*No*" value in the attributes).The different results between its *symmetric* and *asymmetric* representations are appreciable. Figs. 6b and 6c show the results yielded by AT-DC on the Congressional Votes data set on the *symmetric* and *asymmetric* representations, respectively. We observe that classes are better separated in the last representation, at the cost of lower compactness. These results were obtained by exploiting the data ordering described in Section 3.1. In contrast, Fig. 6d shows the results obtained without resorting to data ordering. As we can see, compactness is better, showing that appropriate initializations can further improve the effectiveness of clustering.

Other UCI data sets. The behavior of AT-DC was also evaluated on other data sets from the UCI Repository. The results are summarized in Fig. 7. In particular, we tried the algorithm also on data sets containing a mix of numerical and categorical attributes such as *Horse Colic* and *Labor*. The algorithm was tested both by ignoring and by discretizing the numerical attributes: we adopted both supervised and natural binning discretization. For the latter case, the table reports the results obtained with the optimal number of bins. Within the table, $*$ indicates asymmetric representation, whereas † and ‡ denote supervised and unsupervised discretization, respectively.

Here, the *labor* data set is worth a further discussion. Although the data set is quite small (57 records and 16 attributes, eight of which categorical), the interesting point of the data set is the presence of a large number of missing values. In principle, a tuple exhibiting missing values can be represented as a transaction where all the pairs with missing values are ignored. In general, data sets with a

Dataset	ϵ	Rand	Jaccard	Fowlkes	Γ	No of Clusters
Breast Wisconsin	0.0343	0.8539	0.7377	0.8565	0.7358	13
Horse colic	0.1576	0.5718	0.2730	0.4729	0.2198	18
Horse colic†	0.1711	0.6183	0.3886	0.5750	0.2744	4
Horse colic*,‡ (8 bins)	0.1739	0.6245	0.4120	0.5934	0.2764	3
Credit Approval	0.4261	0.5078	0.3849	0.5579	0.0137	3
Credit Approval†	0.1754	0.6074	0.3068	0.5044	0.2601	5
Credit Approval‡ (8 bins)	0.1768	0.6679	0.4217	0.6108	0.3673	4
Cleveland Heart disease	0.2442	0.5997	0.3686	0.5455	0.2084	3
Cleveland Heart disease†	0.2244	0.6507	0.5124	0.6794	0.3047	2
Cleveland Heart disease‡ (8 bins)	0.2046	0.6566	0.4292	0.6093	0.3282	3
Hungarian Heart disease	0.1871	0.5843	0.3521	0.5363	0.2067	6
Hungarian Heart disease‡ (8 bins)	0.1769	0.7078	0.5851	0.7388	0.4101	2
Hepatitis	0.1806	0.6411	0.5025	0.6884	0.3679	11
Hepatitis†	0.1161	0.6606	0.5215	0.7065	0.4142	13
Hepatitis‡ (4 bins)	0.1484	0.6721	0.5489	0.7220	0.3997	14
Labor	0.2280	0.5169	0.2018	0.3816	0.1003	6
Labor†	0.1578	0.7293	0.6166	0.7642	0.4547	2
Mushrooms	0.0679	0.7261	0.4945	0.6810	0.4897	8
Soybean	0.5974	0.6882	0.1684	0.3621	0.2438	8
Congressional vote	0.1889	0.6928	0.5419	0.7029	0.3853	2
Congressional vote*	0.1014	0.6907	0.4792	0.6607	0.4152	4
Zoo	0.2178	0.9890	0.6600	0.7963	0.7303	5
Zoo*	0.0792	0.9580	0.8437	0.9168	0.8902	7
Advertising*	0.0646	0.4133	0.2577	0.48172	0.1368	14
SMART	0.0115	0.89	0.6773	0.8226	0.7611	40

Fig. 7. Quality indices for selected data sets.

large number of missing values can be assumed to be "intrinsically" transactional, so we expect a good behavior of the algorithm on them. Indeed, we can see that compactness and separability are, in general, good, with a peak when the data set is discretized.

4.4 Synthesized Data

AT-DC was also tested on synthesized data. The data generation process was made parametric to the size N of $\mathcal{M}$ and D of $\mathcal{D}$, the average number T of items for reach transaction and the number C of clusters to force within the data, the percentage ρ of outlier items in $\mathcal{M}$, and the degree ϑ of overlapping among transactions of different clusters.

The synthetic data generation process works as follows: Initially, $\mathcal{M}$ is populated with N items. Then, a subset of items, with the size proportional to ρ, is extracted from $\mathcal{M}$, and C random subsets are generated from the remaining elements in $\mathcal{M}$. Each subset $\mathcal{S}_i$ defines a cluster, and transactions for each cluster are generated starting from such a subset. In particular, a transaction in $\mathcal{C}_i$ is generated by picking its size s from a normal distribution with mean T and fixed variance. Then, the transaction is populated with s items, ϑ percent of which are picked from $\mathcal{S}_i$, and the remaining from the whole $\mathcal{M}$.

In the following experiments, we deliberately chose to ignore data ordering in order to evaluate the potential impact of a random ordering of the data over the performances. Clearly, scalability results when data ordering is considered exhibit a logarithmic worsening, which is due to the initial ordering of the data in the clusters.

4.4.1 Evaluating Scalability

The algorithm was implemented in C++ and executed on an Intel Itanium processor with 4 Gbytes of memory and 2 GHz of clock speed. Fig. 8 reports the performances of the algorithm on different synthetic data sets. In all such figures, N was chosen to be 20 percent of D. As we can see, AT-DC exhibits a practically linear behavior with respect to T, D, and C. This demonstrates that the rate of convergence of both PARTITION-CLUSTER and STABILIZE-CLUSTERS is quite fast. In particular, Fig. 8c details the average number of iterations performed by the PARTITION-CLUSTER procedure with regard to data sets of varying size. (Recall that a single iteration is composed of the instructions P3–P13). Interestingly, the graph exhibits a sublinear relationship between these two variables. Also, notice that the graph depicts both the cases where transactions are initially sorted or unsorted. There seems to be a light improvement in the convergence, even though it is not substantial.

Fig. 8d shows the performances of AT-DC for increasing values of ρ and ϑ (by fixing $D = 100,000$, $C = 4$, and $N = 10,000$). In principle, the ϑ rate has influence on the separability of clusters, whereas the ρ rate has influence on the number of clusters. Thus, it is important to study the performances for these two parameters. Surprisingly, the algorithm exhibits acceptable performances, even when $\vartheta =$ 70 percent (that is, in the presence of highly noisy data). The high values corresponding to values of 80 percent and 90 percent are essentially due to sensible overlap among clusters: In such cases, AT-DC produces several random partitions, which do not reflect the hypothesized structures but exhibit a sort of regularity (due essentially to the data generation process).

Also, there does not seem to exist a neat correlation among the quality of the underlying true classes and the number of iterations performed by the PARTITION-CLUSTERS and STABILIZE-CLUSTERS procedures: Figs. 8e and 8f clearly show that the average number of iterations is bounded by a factor, which only depends on the size of the data.

4.4.2 Evaluating Cluster Quality

Clearly, the ϑ and ρ parameters have a direct influence on the quality of the results of the algorithm. Figs. 9a and 9b

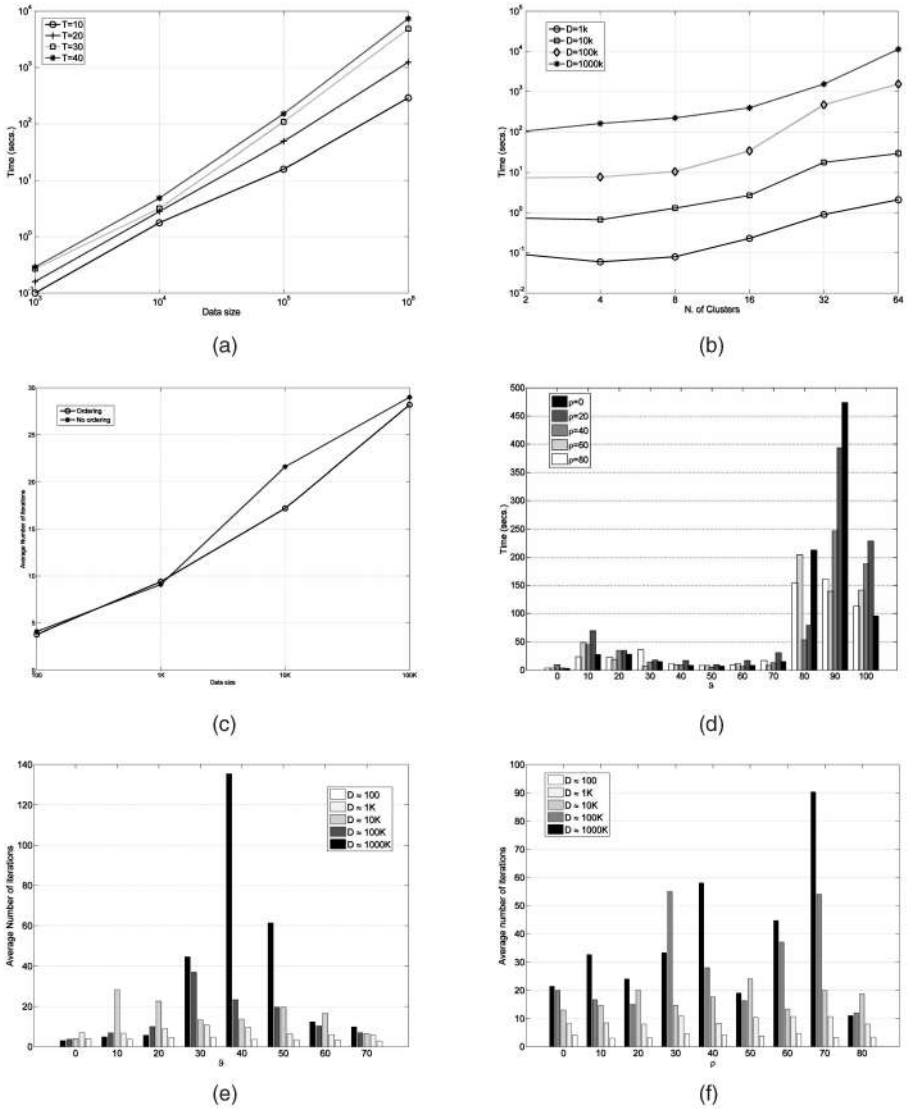


Fig. 8. Cluster scalability. (a) Scalability with respect to D and T $(C = 4, \rho = 0.6, \vartheta = 0.5)$. (b) Scalability with respect to C and D $(T = 10, \rho = 0.6, \vartheta = 0.5)$. (c) PARTITION-CLUSTER: iterations versus data size. (d) Scalability with respect to ρ and ϑ. (e) PARTITION-CLUSTER: iterations versus ρ. (f) PARTITION-CLUSTER: iterations versus ϑ.

show some selected indices describing the overlap and the separation of the clusters. Results were obtained by generating data with fixed values: $D = 100,000$, $T = 10$, $N = 10,000$, and $C = 32$. As we can see, the algorithm is highly tolerant to high values of the two indices. Interestingly, tolerance is higher when $\rho \geq 60$ percent (in particular, if $\vartheta = 50$ percent to 70 percent). Indeed, in such cases, the high degree of overlap, combined with a high number of outlier items, allows the algorithm to discover several clusters, which hence maintain separation on hypothesized clusters.

Fig. 9c shows how the average size T of the transactions and the number N of items available influences the quality of the results. In principle, as N increases, clusters are sparser; that is, the probability of items within clusters tends to decrease. As a consequence, the quality of the resulting partitions tends to decrease as well. However, larger transactions are more tolerant to such a peculiarity (since, of course, item probabilities become more stable).

Finally, Fig. 9d describes the changes in local quality during the invocation of the PARTITION-CLUSTER procedure. Results refer to a data set obtained by fixing $T = 40$, $D = 1,000,000$, $N = 100,000$, and $C = 4$. As we can see, the first three invocations (corresponding to those locating the clusters) achieve a stable quality within less than six steps. The next invocations (concerning rejected splits) require converging less rapidly. Nevertheless, the most significant improvements in quality are located in the very first steps, and the subsequent steps produce improvements of an order of magnitude of 10^{-6}.

The robustness of the algorithm can be appreciated by evaluating and comparing the incidence matrices in Figs. 10a and 10b. Both data sets are generated with $D = 10,000$, $T = 20$, $N = 5K$, $C = 8$, and $\rho = 60$ percent. However, the first one exhibits a degree of overlap $\vartheta = 50$ percent, whereas the second one has $\vartheta = 70$ percent. The analysis of the former plot reveals that there are nine "pure" blocks, that is, that the algorithm found nine clusters

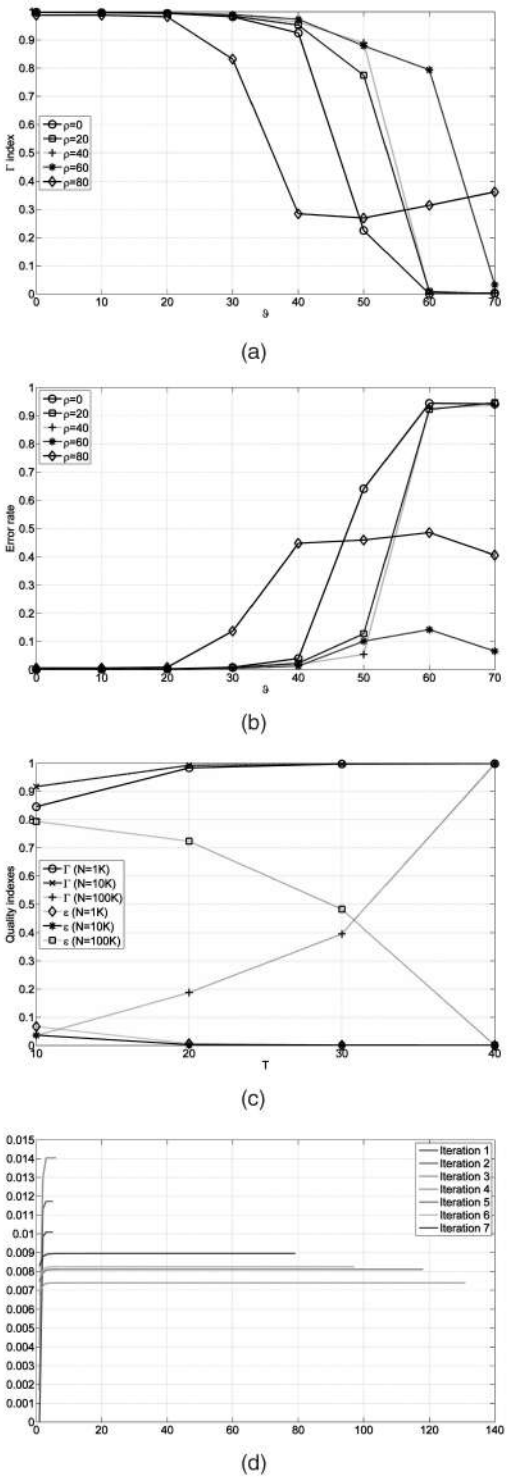


Fig. 9. Quality results for synthesized data. (a) Γ index. (b) Error rate. (c) Quality measures with respect to N and T $(D = 100,000, C = 32)$. (d) PARTITION-CLUSTER: rate of convergence.

(precisely, one more than the actual number of clusters in the data sets). It is worth highlighting that if the degree of transaction overlap is relatively low, each of the discovered blocks exhibits an internal density, which is higher than the one of the outer regions. Clearly, this means that each cluster is inherently characterized by the occurrence of an exclusive set of items. We observe that such an incidence

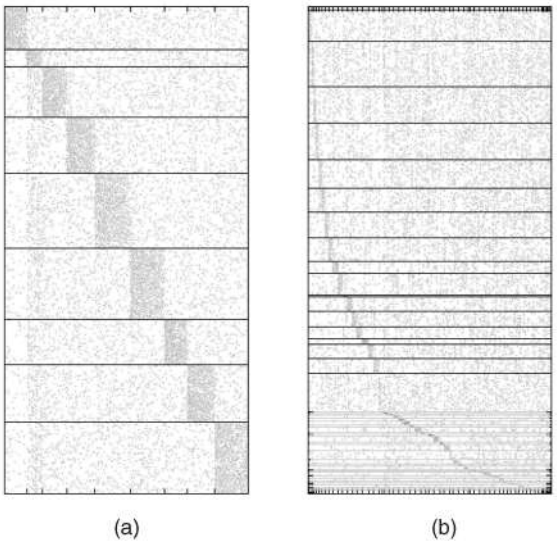


Fig. 10. Transaction/Item incidence matrix for synthesized data sets, with varying degrees of transaction overlap. (a) $\vartheta = 50$ percent. (b) $\vartheta = 70$ percent.

matrix is realistically close to an ideal block-diagonal matrix: this guarantees an excellent quality of the clustering results.

As far as Fig. 10b is concerned, we can notice that AT-DC is still capable of discovering meaningful patterns. Indeed, the high degree of transaction overlap prevents the discovery of the actual clusters. Nevertheless, local regularities are detected and exploited by the algorithm, which thus outputs several small clusters that are strongly characterized by these regularities. This is shown in the plot by the diagonal stripe, which has higher density than the surrounding regions.

4.5 Comparative Analysis

We evaluated AT-DC versus three main algorithms from the current literature, namely, LIMBO, CLICK, and ROCK. CLICK [41] was shown to outperform other approaches adopting a similar hypergraph partitioning strategy [16], [14] and was chosen for comparison, since this claimed to be capable of dealing with high-dimensional categorical data. ROCK [21] is particularly suitable for market-basket data. LIMBO [2], despite its time complexity, was shown to be quite effective.

Fig. 11 summarizes the results of the comparison. We employed some benchmark real-life data sets and four synthetic data sets generated with $T = D = 1,000$, $C = 4$, and $\rho = 40$ percent, and different degrees of overlap ϑ. For the *SMART* data set, we sorted the 14,571 attributes by frequency and considered different versions of the data set, where, for each version, the first d attributes were considered (with d ranging from 50 to 15,000). The results for the LIMBO, ROCK, and CLICK algorithms were obtained by performing an accurate tuning of the input parameters: For each data set, different runs were executed for different values of the parameters, and the best results were chosen. The results shown only refer to the run with the best combination of parameters.

Datasets	AT-DC				LIMBO				CLICK				ROCK			
	ϵ	Γ	*Timing*	$\mathcal{C}$	ϵ	Γ	*Timing*	$\mathcal{C}$	ϵ	Γ	*Timing*	$\mathcal{C}$	ϵ	Γ	*Timing*	$\mathcal{C}$
Congr. Vote	**0.1014**	0.4151	**0.18**	4	0.1241	0.5647	0.81	2	0.2046	0.3439	1.06	6	0.1175	**0.5804**	117	3
Zoo	**0.0792**	**0.8902**	0.19	7	0.0791	0.6867	**0.14**	7	0.1485	0.8262	1.01	7	0.1188	0.9362	6	7
Breast-w	0.0343	0.7358	**1.55**	13	**0.0257**	**0.8989**	2.03	2	0.3491	0.4704	8.18	46	0.041	0.8342	249	5
Mushrooms	0.0679	0.4897	32	8	0.1097	**0.6102**	19.48	2	0.1157	0.3913	**5.54**	11	**0.0039**	0.3993	11466	21
Advertising	**0.0646**	**0.1338**	**16**	14	0.0891	0.1007	1342	8	0.9176	0.0336	70.39	3	0.1399	-0.1128	1871	19
SMART50	0.1472	0.4910	33	36	0.0889	**0.7866**	41.36	3	0.2978	0.5635	**2.5**	3	**0.0865**	0.7608	363	7
SMART100	0.0864	0.6612	75	44	**0.0337**	**0.9102**	127.66	3	0.2231	0.4175	**20.8**	13	0.0638	0.7921	650	10
SMART500	0.0398	0.7115	**152**	40	**0.0128**	**0.9633**	1505	3	-	-	-	-	0.0391	0.8375	769	8
SMART1k	0.0349	0.7373	**261**	46	**0.0056**	**0.9831**	8393	3	-	-	-	-	0.0095	0.9522	2787	9
SMART5k	0.0133	0.7690	**293**	41	**0.0066**	**0.9801**	131387	3	-	-	-	-	0.0111	0.8689	3088	14
SMART10k	0.0128	0.7726	**358**	37	**0.0033**	**0.9897**	242693	3	-	-	-	-	0.0097	**0.8547**	3136	18
SMART15k	0.0115	0.7611	**408**	40	**0.0031**	**0.9857**	539752	3	-	-	-	-	0.0200	**0.8416**	3123	19
Synth, $\vartheta = 40\%$	**0.0602**	**0.8416**	**1**	4	0.0938	0.7678	72	4	-	-	-	-	0.3960	0.1869	12	6
Synth, $\vartheta = 50\%$	**0.5061**	**0.1170**	**2**	7	0.5903	0.0611	80	4	-	-	-	-	0.5639	0.0502	11	11
Synth, $\vartheta = 60\%$	0.6230	0.0142	**2**	11	**0.6163**	0.0315	73	4	-	-	-	-	0.6298	**0.0353**	18	19
Synth, $\vartheta = 70\%$	**0.6893**	**-0.0012**	**2**	8	0.6976	-0.0012	67	4	-	-	-	-	0.6997	-0.0172	19	19

Fig. 11. Comparison among categorical clustering methods ($\mathcal{C} \equiv$ optimal number of clusters).

The evaluation of CLICK poses a number of challenges. First, the algorithm also outputs a *None* cluster, containing all the transactions, which actually do not belong to any clique. For fairness in comparison, we undertook this behavior as a failure in recognizing the hypothesized classes (thus influencing negatively both ϵ and Γ). The poor performance of CLICK on such binary data is due to the fact that the algorithm leverages attribute values to build a graph. Clearly, if there are only two values per attribute, the resulting graph tends to be most likely a complete k-partite graph for some (small) value of its parameters α or $minsup$. This means that there will be only one cluster found, if any. One may resort to selective mining to recover missed clusters. However, in this case, all the 2^d candidate combinations of d items need to be evaluated, which leads to a prohibitive computational cost.

In addition, CLICK did not succeed in satisfactorily clustering *SMART* for values of d that are greater than 100, although the resulting runtimes were acceptable.

AT-DC and CLICK appear to be the best performing algorithms from the efficiency viewpoint. In particular, AT-DC scales well both on the number of tuples and on the dimensionality of the data set, as shown by the experiments on *SMART*. In contrast, ROCK suffers from the size of the data set, whereas LIMBO costs are prohibitive on high-dimensional data (in particular, we were not able to test the algorithm for dimensionalities that are greater than 5,000).

In the first five data sets, AT-DC exhibits quality values that are comparable to those of the competitors. In particular, the *Advertising* data set testifies the good behavior of AT-DC on high-dimensional data with respect to its competitors. Again, on this data set, the low performance of CLICK is due to the high number of transactions belonging to cluster "None."

Concerning the *SMART* collection, LIMBO outperforms the competitors, but the performance of AT-DC progressively increases, as long as dimensionality increases. The latter is a direct consequence of the aforementioned choice of the items. At low dimensionality, items are frequent in all classes and, consequently, the quality function of AT-DC is unable to properly discriminate. On higher dimensionalities, the number of characterizing items for each cluster increases, thus determining a neat improvement. In contrast, competitors seem to suffer from higher dimensionality, since the opposite trend can be observed in ROCK and, apparently, also in LIMBO.

The algorithms were also tested on synthesized data in order to compare their robustness to overlapping clusters. As we can see, all the algorithms behave similarly, but AT-DC seems to exhibit higher robustness. Again, the tuning of CLICK did not provide any significant clustering, even for $\vartheta = 40$ percent.

5 Concluding Remarks

We proposed AT-DC, which is a fully-automatic, parameter-free approach to clustering high-dimensional categorical data. The main advantage of our approach is its capability of avoiding explicit prejudices, expectations, and presumptions on the problem at hand, thus allowing the data itself to speak. This is particularly useful with the problem at hand, where the data is described by several relevant attributes. An intensive experimental evaluation on large data sets showed that the algorithm efficiently yields high-quality results. As a matter of fact, AT-DC exhibits results that are comparable (and, in most cases, even superior) to those of parameter-laden algorithms.

A limitation of the proposed approach is that the underlying notion of cluster quality is not specifically meant for catching conceptual similarities, that is, when distinct values of an attribute are used for denoting the same concept. Indeed, probabilities are leveraged to evaluate cluster homogeneity only in terms of the frequency of items across the underlying transactions. Hence, the resulting notion of quality suffers from the typical limitations of the approaches, which use exact-match similarity measures to assess cluster homogeneity. To this purpose, it is worth noticing that conceptual cluster homogeneity for categorical data can be easily plugged into the framework of the AT-DC algorithm. The idea is learning latent concepts from the data and a collection of characterizing attribute values for each such a concept so that each transaction can be viewed as a mixture of various concepts. Such an approach would enable the discovery of groups of transactions with similar concepts. In this respect, the probability $\Pr(a|\mathcal{C}_k)$ that item a appears in a transaction belonging to cluster $\mathcal{C}_k$ can be replaced by $\Pr(c|\mathcal{C}_k) = \sum_a \Pr(c|a)\Pr(a|\mathcal{C}_k)$, where c denotes a generic (latent) concept. In principle, the identification of three related entities, namely, data attributes, latent concepts, and transaction clusters, lends itself to the employment of a hierarchical probabilistic model for

learning the relationships among such entities. A focus of our ongoing work is adopting the LDA technique [5] to fit our setting.

In addition, since the algorithmic scheme is somehow decoupled from the notion of quality, alternative instantiations can be obtained by adopting different measures, which encode specific application requirements. For example, distinct instantiations can be attempted on different kinds of data such as highly structured data (sequences, graphs, and trees) and time-series data. Interestingly, these data types allow similar definitions of cluster quality, which can be managed in an incremental way. Nevertheless, different assumptions about the model can induce different quality measures on the same categorical data, still enabling the same general scheme of the algorithm. For example, the exploitation of the notion of information loss/gain in terms of entropy would rather produce different clustering, where homogeneity is measured not only in terms of transaction overlap but also proportionally to the absent items. We deserve the investigation of these aspects as a future research direction.

Another promising direction of further research is the extension of the scheme in Fig. 1 in order to deal with outliers. These are transactions whose structure strongly differs from that of the other transactions being characterized by low-frequency items. It is clear that a cluster containing such transaction exhibits low quality. Worst, outliers could negatively affect the PARTITION-CLUSTER procedure by preventing the split to be accepted (because of an arbitrary assignment of such outliers, which would lower the quality of the partitions). Hence, a significant improvement of AT-DC can be obtained by defining an outlier detection procedure that is capable of detecting and removing outlier transactions before partitioning the clusters.

REFERENCES

[1] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '98),* pp. 94-105, 1998.

[2] P. Andritsos, P. Tsaparas, R. Miller, and K. Sevcik, "LIMBO: Scalable Clustering of Categorical Data," *Proc. Ninth Int'l Conf. Extending Database Technology (EDBT '04),* pp. 123-146, 2004.

[3] D. Barbará, J. Couto, and Y. Li, "COOLCAT: An Entropy-Based Algorithm for Categorical Clustering," *Proc. 11th ACM Conf. Information and Knowledge Management (CIKM '02),* pp. 582-589, 2002.

[4] J. Basak and R. Krishnapuram, "Interpretable Hierarchical Clustering by Constructing an Unsupervised Decision Tree," *IEEE Trans. Knowledge and Data Eng.,* vol. 17, no. 1, Jan. 2005.

[5] D.M. Blei, A.Y. Ng, and M.I. Jordan, "Latent Dirichlet Allocation," *J. Machine Learning Research,* vol. 3, pp. 993-1022, 2003.

[6] H. Blockeel, L.D. Raedt, and J. Ramon, "Top-Down Induction of Clustering Trees," *Proc. 15th Int'l Conf. Machine Learning (ICML '98),* pp. 55-63, 1998.

[7] I. Cadez, P. Smyth, and H. Mannila, "Probabilistic Modeling of Transaction Data with Applications to Profiling, Visualization, and Prediction," *Proc. Seventh ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '01),* pp. 37-46, 2001.

[8] M. Carreira-Perpinan and S. Renals, "Practical Identifiability of Finite Mixture of Multivariate Distributions," *Neural Computation,* vol. 12, no. 1, pp. 141-152, 2000.

[9] S. Deerwester et al., "Indexing by Latent Semantic Analysis," *J. Am. Soc. Information Science,* vol. 41, no. 6, 1990.

[10] M. Ester, H.P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," *Proc. Eighth ACM Int'l Conf. Knowledge Discovery and Data Mining (SIGKDD '96),* pp. 226-231, 1996.

[11] D. Fisher, "Knowledge Acquisition via Incremental Conceptual Clustering," *Machine Learning,* vol. 2, pp. 139-172, 1987.

[12] C. Fraley and A. Raftery, "How Many Clusters? Which Clustering Method? The Answer via Model-Based Cluster Analysis," *The Computer J.,* vol. 41, no. 8, 1998.

[13] G. Gan and J. Wu, "Subspace Clustering for High Dimensional Categorical Data," *SIGKDD Explorations,* vol. 6, no. 2, pp. 87-94, 2004.

[14] V. Ganti, J. Gehrke, and R. Ramakrishnan, "CACTUS: Clustering Categorical Data Using Summaries," *Proc. Fifth ACM Conf. Knowledge Discovery and Data Mining (KDD '99),* pp. 73-83, 1999.

[15] A. Gersho and R. Gray, *Vector Quantization and Signal Compression.* Kluwer Academic Publishers, 1991.

[16] D. Gibson, J. Kleinberg, and P. Raghavan, "Clustering Categorical Data: An Approach Based on Dynamical Systems," *VLDB J.,* vol. 8, pp. 222-236, 2000.

[17] A. Gordon, *Classification.* Chapman and Hall/CRC Press, 1999.

[18] C. Gozzi, F. Giannotti, and G. Manco, "Clustering Transactional Data," *Proc. Sixth European Conf. Principles and Practice of Knowledge Discovery in Databases (PKDD '02),* pp. 175-187, 2002.

[19] J. Grabmeier and A. Rudolph, "Techniques of Cluster Algorithms in Data Mining," *Data Mining and Knowledge Discovery,* vol. 6, no. 4, pp. 303-360, 2002.

[20] S. Guha, R. Rastogi, and K. Shim, "CURE: An Efficient Clustering Algorithm for Large Databases," *Proc. ACM SIGMOD Conf. Management of Data (SIGMOD '98),* pp. 73-84, 1998.

[21] S. Guha, R. Rastogi, and K. Shim, "ROCK: A Robust Clustering Algorithm for Categorical Attributes," *Information Systems,* vol. 25, no. 5, pp. 345-366, 2001.

[22] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "Cluster Validity Methods," *SIGMOD Record,* vol. 31, nos. 1-2, 2002.

[23] E. Han, G. Karypis, V. Kumar, and B. Mobasher, "Clustering in a High Dimensional Space Using Hypergraph Models," *Proc. ACM SIGMOD Workshops Research Issues on Data Mining and Knowledge Discovery (DMKD '97),* 1997.

[24] Z. Huang, "Extensions to the K-Means Algorithm for Clustering Large Data Sets with Categorical Values," *Data Mining and Knowledge Discovery,* vol. 2, no. 3, pp. 283-304, 1998.

[25] A. Jain and R. Dubes, *Algorithms for Clustering Data.* Prentice Hall, 1988.

[26] E. Keogh, S. Lonardi, and C. Ratanamahatana, "Towards Parameter-Free Data Mining," *Proc. 10th ACM Conf. Knowledge Discovery and Data Mining (KDD '04),* pp. 206-215, 2004.

[27] B. Liu, Y. Xia, and P. Yu, "Clustering through Decision Tree Construction," *Proc. Ninth Int'l Conf. Information and Knowledge Management (CIKM '00),* pp. 20-29, 2000.

[28] A. McCallum, K. Nigam, and L.H. Ungar, "Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching," *Proc. Sixth Int'l Conf. Knowledge Discovery and Data Mining (KDD '00),* pp. 169-178, 2000.

[29] G. McLachlan and D. Peel, *Finite Mixture Models.* John Wiley & Sons, 2000.

[30] M. Meila and D. Heckerman, "An Experimental Comparison of Model-Based Clustering Methods," *Machine Learning,* vol. 42, no. 1/2, pp. 9-29, 2001.

[31] R. Ng and J. Han, "CLARANS: A Method for Clustering Objects for Spatial Data Mining," *IEEE Trans. Knowledge and Data Eng.,* vol. 14, no. 5, pp. 1003-1016, Sept./Oct. 2002.

[32] M. Ozdal and C. Aykanat, "Hypergraph Models and Algorithms for Data-Pattern-Based Clustering," *Data Mining and Knowledge Discovery,* vol. 9, pp. 29-57, 2004.

[33] L. Parsons, E. Haque, and H. Liu, "Subspace Clustering for High-Dimensional Data: A Review," *SIGKDD Explorations,* vol. 6, no. 1, pp. 90-105, 2004.

[34] D. Pelleg and A. Moore, "X-Means: Extending K-Means with Efficient Estimation of the Number of Clusters," *Proc. 17th Int'l Conf. Machine Learning (ICML '00),* pp. 727-734, 2000.

[35] J.G.S. Zhong, "Generative Model-Based Document Clustering: A Comparative Study," *Knowledge and Information Systems,* vol. 8, no. 3, pp. 374-384, 2005.

[36] P. Smyth, "Model Selection for Probabilistic Clustering Using Cross-Validated Likelihood," *Statistics and Computing,* vol. 10, no. 1, pp. 63-72, 2000.

[37] M. Sultan et al., "Binary Tree-Structured Vector Quantization Approach to Clustering and Visualizing Microarray Data," *Bioinformatics,* vol. 18, 2002.
[38] M.O.T. Li and S. Ma, "Entropy-Based Criterion in Categorical Clustering," *Proc. 21st Int'l Conf. Machine Learning (ICML '04),* pp. 68-75, 2004.
[39] K. Wang, C. Xu, and B. Liu, "Clustering Transactions Using Large Items," *Proc. Eighth Int'l Conf. Information and Knowledge Management (CIKM '99),* pp. 483-490, 1999.
[40] Y. Yang, X. Guan, and J. You, "CLOPE: A Fast and Effective Clustering Algorithm for Transactional Data," *Proc. Eighth ACM Conf. Knowledge Discovery and Data Mining (KDD '02),* pp. 682-687, 2002.
[41] M. Zaki and M. Peters, "CLICK: Mining Subspace Clusters in categorical Data via *k*-Partite Maximal Cliques," *Proc. 21st Int'l Conf. Data Eng. (ICDE '05),* 2005.
[42] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An Efficient Data Clustering Method for Very Large Databases," *Proc. ACM SIGMOD Conf. Management of Data (SIGMOD '96),* pp. 103-114, 1996.

Eugenio Cesario received the PhD degree in systems and computer engineering from the University of Calabria in 2006. From 2003 to 2006, he was a research fellow in the Institute of High-Performance Computing and Networks, National Research Council (ICAR-CNR), Italy, working on data mining systems and applications. He is currently a temporary researcher at ICAR-CNR. His current research interests are distributed data mining, grid programming environments, and grid services architectures.

Giuseppe Manco received the bachelor's degree (summa cum laude) in computer science in 1994 and the PhD degree in computer science from the University of Pisa. He is currently a senior researcher in the Institute of High-Performance Computing and Networks, National Research Council (ICAR-CNR), Italy, and a contract professor at the University of Calabria, Italy. He was a contract researcher at the Centro Nazionale Universitario di Calcolo Elettronico (CNUCE), Pisa, Italy, and a visiting fellow at the Centrum voor Wiskunde en Informatica (CWI), Amsterdam. His current research interests include deductive databases, knowledge discovery and data mining, and Web databases and semistructured data.

Riccardo Ortale received the bachelor's degree (summa cum laude) in computer science engineering from the University of Calabria, the master's degree in Internet software design from the ICT Center of Excellence for Research, Innovation, Education, and Industrial LabsPartnership (CEFRIEL), Politecnico di Milano, in collaboration with Siemens SBS, and the PhD degree in computer science and systems engineering from the University of Calabria. He is currently a researcher in the Institute of High Performance Computing and Networks, National Research Council (ICAR-CNR), Italy, and a contract professor at the University of Calabria. His current research interests include knowledge discovery and data mining, Web databases and semistructured data, and Web personalization.