Artif Intell Rev (2019) 52:1267–1296
https://doi.org/10.1007/s10462-018-9627-1

A review of conceptual clustering algorithms

Airel Pérez-Suárez[1] · **José F. Martínez-Trinidad**[2] · **Jesús A. Carrasco-Ochoa**[2]

Published online: 23 March 2018

Abstract Clustering is a fundamental technique in data mining and pattern recognition, which has been successfully applied in several contexts. However, most of the clustering algorithms developed so far have been focused only in organizing the collection of objects into a set of clusters, leaving the interpretation of those clusters to the user. Conceptual clustering algorithms, in addition to the list of objects belonging to the clusters, provide for each cluster one or several concepts, as an explanation of the clusters. In this work, we present an overview of the most influential algorithms reported in the field of conceptual clustering, highlighting their limitations or drawbacks. Additionally, we present a taxonomy of these methods as well as a qualitative comparison of these algorithms, regarding a set of characteristics desirable since a practical point of view, which may help in the selection of the most appropriate method for solving a problem at hand. Finally, some research lines that need to be further developed in the context of conceptual clustering are discussed.

Keywords Data Mining · Clustering · Conceptual clustering · Concept formation

1 Introduction

Clustering is an important Data mining and pattern recognition technique. This technique aims at organizing a collection of objects into classes or *clusters*, such that objects belonging to the same cluster are similar enough to infer they are of the same type; and objects belonging

✉ Airel Pérez-Suárez
asuarez@cenatav.co.cu

José F. Martínez-Trinidad
fmartine@inaoep.mx

Jesús A. Carrasco-Ochoa
ariel@inaoep.mx

[1] Advanced Technologies Application Center (CENATAV), Havana, Cuba

[2] National Institute of Astrophysics Optics and Electronics (INAOE), Puebla, Mexico

to different clusters are dissimilar enough to infer they are of distinct type (Pfitzner et al. 2009). There are several areas in which clustering algorithms have been successfully applied, like for instance: Moving Object Activity Prediction (Yuan et al. 2017), Medicine (Michel et al. 2012), community detection (Wang et al. 2016), Bug prioritization (Uddin et al. 2017) and fingerprint analysis (Munir et al. 2012), among others.

Clustering algorithms can be classified according to different criteria (Jain et al. 1999), for example: (a) the type of clusters the algorithm builds, (b) how the algorithm deals with changes in an already clustered collection, (c) the type of organization the algorithm builds with the objects, and (d) the approach followed by the algorithm in order to build the clusters, among other criteria. This work focuses its analysis on the last one criterion. Regarding to this criterion, a clustering algorithm can be classified as those algorithms that build the clusters based on the similarity relations existing among the objects of the collection, but they do not provide an explanation about the obtained clusters; and those algorithms which build a set of clusters based on concepts or properties, such that the result they provide is a set of clusters, together with the intentional description of each one of these clusters; these algorithms are usually known as *conceptual clustering algorithms*. Most of the clustering algorithms developed so far are based on the similarity relations existing between the objects and they leave to the specialist the hard task of interpreting or explaining the obtained clusters; these algorithms could be not suitable for applications in which the users do not only wish to know the clusters existing in the data but some additional information, in terms of the variables of the problem, that explains why certain clusters are formed.

There are several conceptual clustering algorithms reported in the literature. Techniques like biclustering, formal concept analysis, lattice computing or frequent pattern extraction, among others, have been used for developing conceptual clustering algorithms. Despite the achievements of the existing algorithms, most of them have several limitations that could make it difficult their application in real problems. These limitations are mainly related with a high computational complexity (e.g., algorithms LABYRINTH, FLOC and INCOSHAM, among others), the needed of tuning several parameters (e.g., algorithms CPC, SAMBA and F^2IHC, among others), the inefficient updating of the clustering after changes in the collection (e.g., algorithms FTC, GCC and SDHC, among others), the high complexity of the language used for building the concepts (e.g., algorithms CTWC, LINNEO+, and COBWEB, among others) and the impossibility of processing objects described by mixed and incomplete data (e.g., algorithms DF, TDC, and WebDCC, among others); these drawbacks make the study of conceptual clustering algorithms a research area of concern nowadays. Hereinafter, we will talk indistinctly about conceptual clustering algorithm and conceptual algorithm to refer to the same term.

The aim of this work is to present an overview of the most influential conceptual clustering algorithms reported in the literature, describing their main characteristics and highlighting their limitations since a practical point of view.

The structure of this paper is as follows: in Sect. 2 the most significant conceptual clustering algorithms of the state of the art are described. A qualitative comparison of the previously described algorithms, regarding some important characteristics that should be present in a conceptual clustering algorithm, is discussed in Sect. 3 whilst Sect. 4 describes some applications of conceptual clustering algorithms in real problems. Finally, in Sect. 5 some research lines that need to be further developed in the context of conceptual clustering are presented.

2 Conceptual clustering algorithms

Although before 1970 the term concept formation had been already used, the authors concerning with the study of conceptual clustering algorithms acknowledge as the pioneer of this research line to Ryszard S. Michalski, from his works with the CLUSTER/PAF algorithm (Michalski 1980) in the eighties. Formally, the problem of conceptual clustering is defined as follows.

Let $O = \{o_1, o_2, \ldots, o_n\}$ be a collection of objects, each one described in terms of a set of attributes $A = \{a_1, a_2, \ldots, a_m\}$, which can be quantitative or qualitative. These features can also be univalent, if they get only one value from the set of all their possible values, or multivalent, if they can get more than one value at the same time; i.e., their value is a set of values. Additionally, there are problems in which the values of some features are unknown for one or more objects; i.e., there exist *missing data*.

The problem of conceptual clustering consist in organizing the collection O into a set of clusters $G = \{g_1, g_2, \ldots, g_k\}$, such that the following three conditions are fulfilled:

- each cluster $g_i, i = 1, \ldots, k$ has associated an extensional and an intentional description; the extensional description contains the objects that belong to the cluster whereas the intentional one contains the concepts (statistical or logical properties) that describe the objects belonging to the cluster, in terms of the attribute set A and using a given language. Depending on the complexity of the language the concepts will be difficult or easy to understand by a final user.
- $\forall o_i \in O, i = 1, \ldots, n, \exists g_j \in G, j = 1, \ldots, k$ such that $o_i \in g_j$. This property ensures that every object in the collection belongs to at least one cluster.
- $\nexists g_j \in G, j = 1, \ldots, k$ such that $g_j = \emptyset$. This property ensures that each cluster in the solution contains at least one object of the collection.

There are several conceptual clustering algorithms that, in addition of generating the clusters and their concepts, they build hierarchies or lattices using the properties that fulfil these concepts. Moreover, the clusters built by these algorithms can be disjoint, overlapping or fuzzy, depending on the belonging of the objects to the clusters. Additionally, these algorithms are incremental if they update the clustering when new objects are added to the collection,by using the current clusters; or they are static if every time the collection changes they need to re-process the whole collection again in order to update the clustering.

The existing conceptual clustering algorithms generally follow one of the following approaches: (i) they build first the extensional description of the clusters and based on that, they build after the intentional description of each cluster; or (ii) they build first a set of concepts from the collection and then, based on these concepts, they build the extensional description of each cluster.

For a better comprehension of the algorithms we will analyze in this section, we have organized them into a taxonomy based on the main paradigm they use for building the conceptual clustering. The taxonomy of the reviewed methods is showed in Fig. 1.

2.1 Optimization based algorithms

Here, we review those conceptual algorithms that, starting from an initial set of clusters, they build a clustering through a continuous process of optimization of this initial set, generally by using an objective function.

CLUSTER/PAF (Michalski 1980) is an algorithm following an optimization approach. Given a predefined k number of seed objects, this algorithm builds first a set of k l-complexes,

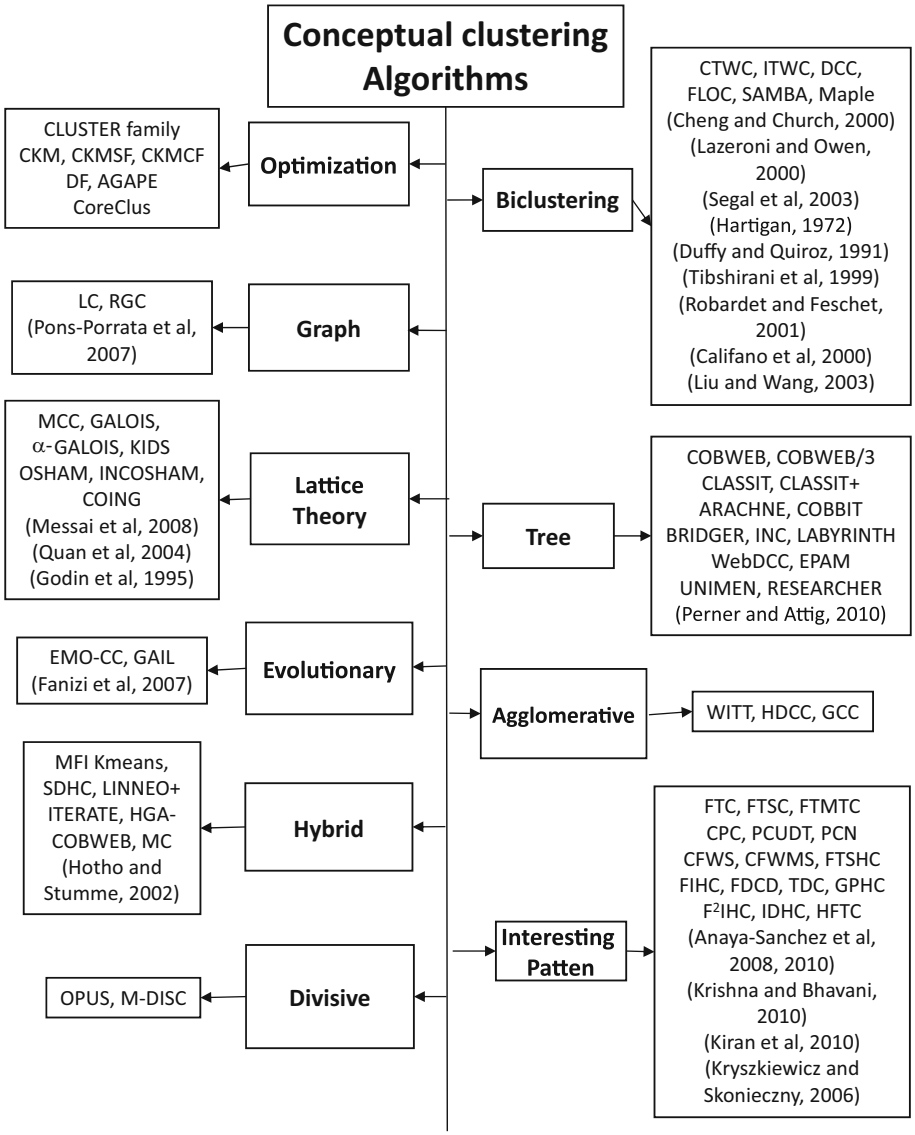

Fig. 1 Proposed taxonomy of the conceptual algorithms analyzed in this work

representing a set of concepts existing in the collection. An l-complex is the logic product of substructures of the type $[a_j \# v]$, where $a_j \in A$ is a feature describing an object, v is a value in the domain of a_j, and # is an operator in $\{\neq, \leq, <, >, \geq, =\}$. Afterwards, these l-complexes are iteratively modified through a process such that the set of sets of objects covered by each resulting l-complex constitutes a disjoint clustering of the data. The aforementioned process is iteratively repeated as long as a predefined evaluation function improves. Reported extensions of this algorithm are CLUSTER/2 (Michalski and Stepp 1983), CLUSTER/S (Stepp and Michalski 1986) and CLUSTER/3 (Seeman and Michalski 2006) algorithms. CLUSTER/2 introduces improvements in the way in which both the concepts and the disjoint clusters are

obtained. CLUSTER/S applies the same strategy of CLUSTER/2 but it allows to include external knowledge in the clustering process. Finally, CLUSTER/3 applies CLUSTER/PAF iteratively in order to build a hierarchy of clusters.

CKM (Ralambondrainy 1995) and W-Kmeans (Bouras and Tsogkas 2010) are conceptual algorithms based on the classical kmeans algorithm, whilst CoReClus (Fonseca et al. 2012) is based on the Kmedoids algorithm. CKM first transforms the qualitative features describing the objects into quantitative ones and then it uses the Kmeans algorithm for building the clustering. After that, CKM transforms the quantitative features describing the objects into qualitative ones and it builds the description of each cluster by searching for a logic predicate that generalizes each object in the cluster. Reported extensions of CKM are CKMSF (Ayaquica-Martínez et al. 2005) and CKMCF (Ayaquica-Martínez et al. 2006) algorithms. CKMSF uses the KMSF algorithm (García-Serrano and Martínez-Trinidad 1999) for building the clusters and it uses a different approach for transforming the features; CKMCF builds the clusters following the ideas of CKMSF but it changes the way in which the concepts are generated. On the other hand, W-Kmeans first uses WordNet (Miller 1995) and kmeans for building the clusters and then, it builds the concepts describing each cluster using WordNet. Finally, CoReClus uses the Kmedoids algorithm for building a set of disjoint clusters; the concept describing each cluster is a set of logic clauses extracted from the objects using Inductive Logic Programming and a given background knowledge.

Other algorithms following an optimization approach are DF (Velcin and Ganascia 2007a) and AGAPE (Velcin and Ganascia 2007b). DF starts with a randomized initial set of *stereotypes* and then, it iteratively improves this set using the TABU search meta-heuristic (Ng and Wong 2002) as long as a predefined evaluation function improves. A stereotype is a set of feature-value pairs that, unlike the concept of prototype, it does not have to coincide with the typical or "average"" object in the cluster. Once the final set of stereotypes is found, the clusters are built using the set of objects covered by each stereotype. AGAPE follows the same principles of DF but it uses different heuristics operators during the TABU search.

All the algorithms described in this section require to tune up several parameters (i.e., the number of clusters or some thresholds used during the clusters formation process) which could depend on the collection to process. In general, users do not have any a priory knowledge about the collection they want to cluster; therefore, to tune up several parameters could be a difficult task. Besides CLUSTER/PAF, CLUSTER/2, CLUSTER/S, CLUSTER/3, CKMCF, DF and AGAPE algorithms have a high computational complexity that can get to be exponential for the algorithms of the CLUSTER family. Additionally, CKM, CKMSF, CKMCF and CoReClus build clusters whose concepts are difficult to understand in real problems, either because they are very large or because the theory or the description language used to build them is inherently difficult to understand. DF and AGAPE algorithms require a lot of computer memory and they can build a large number of clusters. Usually, in real problems the number of clusters is a priori unknown; however, if the number of clusters grows, analyzing those clusters could be as difficult as analyzing the entire collection. Finally, CKM and CKMSF, as well as CoReClus, require external knowledge which could not be always available in real problems.

2.2 Tree based algorithms

In this subsection, we describe those conceptual algorithms that use a classification tree or a hierarchy-like structure, for organizing the objects. Each node in this structure represents a class and it is labeled by a concept that describes the objects classified under the node.

COBWEB (Fisher 1987), is an well-known incremental and hierarchical conceptual algorithm that describes the clusters using pairs of attribute-value which summarize the probability distribution of objects classified under the sub-tree determined by that cluster. For processing each object, COBWEB recursively searches the hierarchy, using a function called *Category Utility* (CU, for short), looking for the most suitable node in which the object can be placed. Through this search, COBWEB updates the concepts of the visited nodes and it computes, using a function called *Category Utility* (CU, for short), the quality of the partitions induced by: (a) including the object in each child node of the current node, (b) merging the nodes forming the two best induced partitions, (c) adding the object as a new child node, or (d) splitting the node forming the best induced partition; the partition having the highest quality represents the action COBWEB will perform over the hierarchy. This recursive search continues through the respective node containing the object in the selected best partition, until a terminal node is reached. In such case, a new node is created for containing the new object.

Known extensions of COBWEB are CLASSIT (Gennari et al. 1989), CLASSIT+ (Sahoo et al. 2006), COBWEB/3 (McKusick and Thompson 1990), DynamicWEB (Scanlan et al. 2008), ARACHNE (McKusick and Langley 1991), COBBIT (Kilander and Jansson 1993) and BRIDGER (Reich and Fenves 1991). CLASSIT extends COBWEB by modifying the CU function in order to be able to process objects described by numerical features. CLASSIT+ is a variant of CLASSIT that defines a new CU function taking into account the katz distribution (Katz 1996). COBWEB/3 combines the methodologies used in COBWEB and CLASSIT algorithms, for being able to process objects described by mixed features. DynamicWEB extends COBWEB for being able to process deletions and modifications of objects; modifications are understood as a deletion following by an addition. ARACHNE extends COBWEB by adding, after the updating of the hierarch, the verification of two conditions which aim to reduce the data order dependence of original COBWEB. On the other hand, COBBIT extends COBWEB, by using a time window, for improving its scalability, the maintenance of concepts, and the data order dependency. Finally, BRIDGER extends COBWEB for processing objects described by mixed features; BRIDGER uses the CU function defined by COBWEB/3 but it modifies the searching process performed for allocating a new object in the hierarchy.

Another algorithms under this paradigm are INC (Hadzikadic and Yun 1989), WebDCC (Godoy and Amandí 2006), EPAM (Feigenbaum 1961) and UNIMEN and RESEARCHER (Lebowitz 1986). These algorithms are incremental and they process each object by searching, in the hierarchy-like structure they build, for a node fulfilling a specific property or optimizing a predefined function; once this node is found, they can: (a) inserting the object into that node as a child node or (b) crating a new node for storing the object. INC uses a similarity function for guiding the searching process and different from COBWEB, INC describes the nodes using a set of pairs attribute-value, which have associated a value representing the relevance of each pair for describing the objects contained in the node. WebDCC looks for the deepest node whose similarity with the document exceeds a predefined threshold value and it uses a variant of the *simple-pass* algorithm (Hill 1968) for adding the document to the clusters contained in the node; this variant can create new nodes or it can merge or splitting existing ones depending on their usefulness for describing the documents they contain. EPAM and UNIMEN use the same strategy than WebDCC for looking for the node but they use a matching function and a distance function, respectively, instead of a similarity one. RESEARCHER is a version of UNIMEN developed for processing objects described by both attributes and relations between those attributes.

LABYRINTH (Langley et al. 1990; Thompson and Langley 1991) is an incremental algorithm that builds a hierarchy of concepts. Unlike previous algorithms, LABYRINTH assumes

the objects are described by a set of components that can be primitives (i.e., comprised only of qualitative values) or structured (i.e., comprised of other component). For processing an object, LABYRINTH uses a variant of COBWEB called COBWEB' for adding each one of its components to the hierarchy. Finally, once all the components were added, the object itself is added to the hierarchy using COBWEB'.

Perner and Attig propose in Perner and Attig (2010) an incremental and hierarchical fuzzy conceptual algorithm. This algorithm follows the same general idea of COBWEB (Fisher 1987) but it uses a score function instead of the CU function. Additionally, this algorithm can place an object into more than one node and it describes the clusters using a fuzzy centroid computed from the objects stored in the cluster and their membership values.

Excepting WebDCC and EPAM all the other algorithms build clusters whose concepts can be difficult to understand in real problems because they are very large or because the theory or the description language used to build them is inherently difficult to understand. Moreover, excepting COBBIT, INC and WebDCC, all the remaining algorithms have a high computational complexity which makes them unsuitable for applications dealing with a large number of objects. Besides, all the algorithms have to process the objects one by one so they could be less useful in such contexts in which data changes frequently. Additionally, UNIMEN, RESEARCHER, INC, WebDCC and the algorithm proposed in Perner and Attig (2010) require to tune up two or more parameters used for building the clusters, which depend on the collection to process. Generally, users do not have any a priory knowledge about the collection therefore, tuning up several parameters could be a difficult task. Finally, excepting DynamicWEB, ARACHNE, COBBIT and INC, the other algorithms depend on data order analysis so they can build a different hierarchy for the same collection, each time the algorithm is executed.

2.3 Graph based algorithms

In this section, we review those algorithms that use graph theory for building the clusters. Generally, these algorithms represent the collection of objects as a graph, in which vertices are the objects and edges express the similarity relationship existing between objects, and they build a clustering through a covering of this graph using a special kind of subgraph.

Examples of algorithms based on this approach are LC (Martínez-Trinidad and Sánchez-Díaz 2001), RGC (Pons-Porrata et al. 2002b) and the algorithm reported in Pons-Porrata et al. (2007). LC and RGC propose to use as subgraphs the connected components or the *compact sets*, see Martínez-Trinidad and Sánchez-Díaz (2001) for an explanation of these concepts; in this context, each subgraph constitutes a cluster. Once the clustering is built, LC builds as concept of each cluster the disjunction of the l-complexes built from the *typical testors* of the cluster. A typical testor is a subset of features that discerns between the objects of the cluster and the remaining clusters. On the other hand, RGC builds as concepts only the *exclusive l-complexes*; i.e., those l-complexes covering only the objects included in the cluster. Finally, the algorithm reported in Pons-Porrata et al. (2007) assumes each news constitutes a cluster and then it iteratively builds each level of a hierarchy by applying the IC graph-based algorithm (Pons-Porrata et al. 2002a), over the set of centroids of the clusters of the previous level. The concept describing a cluster is conformed by the sentences that contain the highest number of typical testors of the cluster.

LC and RGC, as well as the algorithm proposed in Pons-Porrata et al. (2007), have a high computational complexity that can get to be exponential. Additionally, these algorithms can build clusters described by a large number of terms, which can be also difficult to understand in real problems. Finally, LC algorithm can leave clusters without a description and the

algorithm introduced in Pons-Porrata et al. (2007) builds hierarchies that can be difficult to browse due to the large number of clusters and levels they have.

2.4 Agglomerative algorithms

In this subsection we describe the algorithms that follow a bottom-up approach for building the clustering. The algorithms based on this approach generally perform the following steps: (a) assuming each object constitutes a cluster; (b) selecting a pair of clusters fulfilling a property and/or optimizing a predefined function; (c) merging the clusters selected in step (b); usually, steps (b) and (c) are repeated until a predefined stop condition is fulfilled.

Examples of algorithms based on this approach are WITT (Hanson and Bauer 1989), HDCC (Funes et al. 2008) and GCC (Talavera and Béjar 2001).

WITT selects in step (b) the closest pair of clusters, taking into account a distance function based on information loss, and it stops the building process when this distance is greater than a predefined threshold. WIIT describes the clusters using a set of *contingency tables*, one for each possible pair of attributes describing the objects of the cluster and it performs a post-processing on remaining singleton clusters; a variant of WITT which proposes another post-processing of the singleton clusters was proposed in Talmon et al. (1993). HDCC follows the idea of WITT in step (b) but it stops the hierarchical construction process when all the objects belong to a single cluster. Additionally, HDCC assumes the user supply two functions for obtaining the concepts describing the clusters and it can also merge in each level all those other clusters containing objects covered by the concept describing the two merged clusters. Finally, GCC describes the clusters in the hierarchy using the same theory as COBWEB and it selects in step (b) the cluster having the less *level of generality* and its most similar cluster; the level of generality of a cluster is a property that expresses the predictability and predictiveness of the pairs attributes-value that compose the concept describing that cluster.

The algorithms described in this section build clusters whose concepts can be difficult to understand in real problems because they are very large or because the theory used to build them is inherently difficult to understand. Besides, WITT requires to tune up the values of three parameters used for building the clusters, whose values depend on the collection to process. As it was mentioned before, to tune up values for several parameters could be a difficult task to perform in real problems. On the other hand, HDCC algorithm has a high computational complexity that makes it less useful in real problems dealing with a large number of objects. In addition, HDCC can build clusters whose concepts do not describe all the objects in the clusters; this can be a problem in real applications.

2.5 Divisive algorithms

In this subsection we describe the algorithms that follow a top-down approach for building the conceptual clustering. The algorithms following this approach generally perform the following steps: (a) assuming as the top level of the hierarchy the cluster comprised of all the objects; (b) splitting each node into two or more clusters by using a state-of-the-art clustering algorithm or by following a specific partitioning strategy. Usually, these steps are repeated a predefined number of times or until a predefined stop condition is fulfilled.

Examples of clustering algorithms based on this approach are OPUS (Nordhausen 1986) and M-DISC (Chu et al. 1996), which describe a cluster using the set of attribute-value pairs existing in the path from the root to the cluster. For partitioning a node in step (b), OPUS selects the best partition, according to a predefined evaluation function, among all the partitions induced by each feature describing the objects. M-DISC (Chu et al. 1996) follows

a strategy similar to that used by the Bisecting Kmeans algorithm (Zhao and Karypis 2002) but it uses a variation of the CU function defined by COBWEB (Fisher 1987), in order to compute the quality of all possible binary partitions.

Both M-DISC and OPUS have a high computational complexity and they build clusters described by a large number of terms; these two characteristics make these algorithms less useful in real problems dealing with a large number of objects. In addition, the hierarchies built by M-DISC could have a large number of clusters and levels, and they depend on the order the objects arrive; these characteristics make these hierarchies difficult to browse in real problems. Finally, OPUS needs to know in advance a set of binary relations between the objects that does not necessarily have to be available in all real problems.

2.6 Interesting-pattern based algorithms

In this subsection, we review the conceptual algorithms that build the clustering based on a subset of *patterns* they discover from the set of objects, each one meeting a criterium that makes it *interesting* from a point of view. Generally, a pattern is a structure that belongs to the description of an object. These structures can be sets, sequences, trees or graphs, among others. The set of objects in which a pattern is presented and the size of this set are known as the *cover* and the *support* of the pattern, respectively. The interestingness of a pattern is measured by a function that could be as simple as its frequency of apparition in the data. Taking into account this criterium, a pattern is considered as *frequent* (i.e., interesting) in a set of objects iff its support is greater than a predefined threshold value.

The algorithms following this approach assume each extracted pattern and its cover constitutes the concept and the cluster, respectively. Most of these algorithms were proposed for processing documents or news and they follow a common framework. The steps used by those non hierarchical pattern-based algorithms are: (a) preprocessing the objects for removing low discriminative features; (b) extracting the interesting patterns, generally by using state-of-the-art algorithms, a special data structure, or another strategy; (c) selecting a subset of the extracted patterns (i.e., clusters) according to the fulfilment of one or more properties or based on the value of a predefined function; and (d) reducing the number of the detected clusters by merging highly similar clusters or by using a predefined function and a threshold.

Examples of algorithms following the above mentioned steps are FTC (Beil et al. 2002), FTSC (Liu and he 2005), the algorithm proposed in (Krishna and Bhavani 2010), CPC (Fore and Dong 2012), PCUDT (Gutiérrez-Rodríguez et al. 2015b), PCN (Gutiérrez-Rodríguez et al. 2015a), FTMTC (Zheng et al. 2014), the algorithms CFWS and CFWMS reported in (Li et al. 2008), as well as the algorithm proposed in (Anaya-Sánchez et al. 2008).

FTC uses the Apriori algorithm (Agrawal et al. 1993) for extracting frequent itemsets in step (b), and it considers four functions, reported in Shi and Ester (2003) and based on the overlapping of the clusters, as alternatives for step (c). FTSC is the same as FTC but considering only two of the four functions proposed in Shi and Ester (2003). CFWS uses a *generalized suffix tree* for extracting *frequent sequences of items* in step (b). On the other hand, CFWMS includes also in step (a) the substitution of verbs and nouns by their related concepts of WordNet (Miller 1995) and their synonyms, and consequently it extracts *frequent sequences of concepts* using the same structure as CFWS. FTMTC also uses frequent itemsets but it proposes to use, in step (c), a function which measures the quality of a cluster based on its size, the size of the itemset describing the cluster and the relationship between the objects and the itemset. The algorithm reported in Krishna and Bhavani (2010) uses also frequent itemsets but instead of reducing the number of clusters in step (d), it enriches the concept describing each detected cluster by adding some terms describing the documents in

the clusters. The algorithm reported in Anaya-Sánchez et al. (2008) does not perform the step (d) and from all the itemsets of size 2 having the highest probability of co-occurring in the collection, extracted in step (b), it only keeps in step (c) those such that the cohesion of the set of documents they cover surpass a predefined threshold; a version of this algorithm that include more term in the generated concepts was proposed in Anaya-Sánchez et al. (2010). Finally, CPC, PCUDT and PCN uses *contrasting itemsets* in step (b); a pattern is contrasting if it has a high occurrence in a specific cluster and a low occurrence in the remaining ones. These three algorithms follow the same ideas with the difference that CPC extracts these patterns using the FP-growth (Han et al. 2004) algorithm and a heuristic whilst PCUDT and PCN use a collection of unsupervised decision trees. Instead of reducing the number of clusters, in step (d) these algorithms perform a strategy which aims to reassign the objects to the cluster with the highest value of a predefined function.

In addition of the aforementioned steps a,b,c and d, almost all the hierarchical algorithms based on this approach perform the following steps: (e) arranging the detected clusters into a hierarchy based on the subset or superset relations existing among the patterns describing the clusters or by using another predefined relation among these patterns; (f) reducing the number of clusters in which an object can be included by using a specific heuristic or a predefined membership function and a threshold; and (g) reducing the number of clusters of the top level of the hierarchy by merging those highly similar nodes (i.e., clusters) or using a state-of-the-art clustering method.

Examples of hierarchical conceptual algorithms following the aforementioned steps are FTSHC (Wang and Liu 2011), FIHC (Fung et al. 2003), FCDC (Baghel and Dhir 2010), TDC (Yu et al. 2004), GPHC (Malik and Kender 2006), F^2IHC (Chun-Ling et al. 2010), IDHC (Malik et al. 2010), and the algorithms reported in (Kryszkiewicz and Skonieczny 2006) and (Kiran et al. 2010).

FTSHC uses the FTSC algorithm (Liu and he 2005) for building the clusters and it only performs the above mentioned step (e). FIHC uses frequent itemsets extracted by using the Apriori algorithm and, in addition to step (g), it iteratively merges those nodes having a parent relationship whose similarity surpass a predefined threshold. FCDC is the same as FIHC but it preprocesses the collection in step (a) by using WordNet (Miller 1995). TDC uses *closed itemsets* computed using the CLOSET+ algorithm (Wang et al. 2003) and it uses the UPGMA algorithm (Jain et al. 1999) in step (g); an itemset is closed iff it is frequent and its support is different from the support of each of its superset itemsets. The algorithm proposed in Kryszkiewicz and Skonieczny (2006) is also the same as FIHC but using closed itemsets. On the other hand, GPHC uses *closed and interesting itemsets* for building the clusters and it employs the bisecting K-means algorithm (Zhao and Karypis 2002) in step (g). A closed itemset is interesting if its interestingness, measured using a predefined function, surpass a threshold; GPHC proposes five alternatives for measuring the interestingness of a itemset. F^2IHC follows the ideas of FIHC but it uses frequent and fuzzy itemsets, extracted by using a varian of the Apriori algorithm, and it does not perform the step (g). IDHC uses *sigficant* itemsets of size 2 for building the clusters and, like GPHC, it uses the bisecting K-means algorithm in step (g). Finally, the algorithm reported in Kiran et al. (2010) combines the ideas of the TDC (Yu et al. 2004) and GHPC (Malik and Kender 2006) algorithms and after step (g), it enriches the concepts of the clusters by using Wikipedia.

HFTC (Beil et al. 2002) builds a conceptual hierarchy in which the frequent itemsets of size 1 and theirs covers constitute the top level of the hierarchy. Starting from this point, HFTC builds the remaining levels by partitioning each cluster using the above described FTC algorithm.

The algorithms FTC, HFTC, FTSC, FTSHC, CFWS, CFWMS, FTMTC, FIHC, as well as FCDC and the algorithm proposed in Kryszkiewicz and Skonieczny (2006), depend on the value of the threshold used for building the frequent patterns. Low values of this parameter could make these algorithms computational expensive or even intractable, while high values could make the algorithms to produce clusters of low quality due to the missing of interesting frequent patterns. On the other hand, TDC, GPHC, F^2IHC, IDHC, CFWS, CFWMS, FTMTC, CPC, PCUDT and the algorithms proposed in Kiran et al. (2010), Krishna and Bhavani (2010) need to tune up the value of two or even more parameters used for building the clusters, which can be difficult to set up in real problems. FIHC, FCDC, F^2IHC, CPC and the algorithms in Kryszkiewicz and Skonieczny (2006), Kiran et al. (2010) have both high computational complexity and they requiere a lot of computer memory, whilst CFWS and CFWMS also requieres a lot of computer memory. These characteristics can make these algorithms less useful in real problems. Additionally, CPC and GHPC do not guarantee that all the objects belong to at least one cluster. Finally, the algorithm proposed in Kiran et al. (2010) as well as CFWS and CFWMS can build clusters having large descriptions, which could be difficult to understand in real problems.

2.7 Evolutionary algorithms

In this subsection, we review the conceptual clustering algorithms that build the clusters using an *evolutionary* approach. These algorithms generates solutions using techniques inspired by nature, such as *inheritance*, *mutation*, *selection* and *crossover*.

The general framework of the algorithms reported in this section includes the following steps: (a) generating an initial population of individuals (i.e., solutions of the problem at hand) taking into account a predefined representation; (b) applying the evolutionary operators over the current population for building the next generation; (c) evaluating the current and next generations using one or several predefined objective functions; and (d) applying a predefined heuristic for keeping and improving the best solutions found so far. Usually, these steps are repeated a predefined number of times or until a stop criterium is fulfilled.

Examples of algorithms following this approach are EMO-CC (Romero-Zaliz et al. 2008), GAIL (Han et al. 1997) and the algorithm reported in Fanizzi et al. (2007). EMO-CC uses a variation of the NSGA-II (Deb et al. 2002) algorithm in order to search for a set of substructures in a graph database. Each resulting substructure as well as the set of observations in which the substructure occurs constitutes a concept and the cluster. GAIL codifies each individual as a set of attributes-value pairs and it uses the classical evolutionary approach together with three objective functions it proposes; the attribute-value pair in each resulting individual constitute the concept whilst its cover constitute the cluster. Finally, the algorithm proposes in Fanizzi et al. (2007) uses an evolutionary variation of Kmedoids for building a set of disjoint clusters whose concepts are after built by using the features describing the objects and a specific description logic.

The GAIL and EMO-CC algorithms need to tune up three or more parameters used for building the clustering, which depend on the collection to process. On the other hand, the algorithm proposed in Fanizzi et al. (2007) can build clusters having a large descriptions and it has a high computational complexity.

2.8 Biclustering algorithms

In this subsection, we briefly describe the algorithms that conduct a *biclustering*; that is, they do a clustering in the set of objects as well as in the set of features describing those objects.

These two clusterings satisfy that any of them can be obtained from the other one; this way, the cluster of features associated to a cluster of objects can be seen as the concept describing those objects. Names as *coclustering*, *bidemensional clustering*, *subspace clustering* and *block clustering*, among others, are usually used in the literature to refer to the biclustering problem.

Most of the algorithm reviewed in this section assume the collection of objects is represented using a numerical matrix where the rows and columns represent the objects and the features describing these objects, respectively. Thus, the main idea of biclustering algorithms is to search for a set of sub-matrixes (i.e., biclusters) fulfilling a specific property. There are two main approaches for building these biclusters.

The first approach is to discover them one by one, generally performing the following steps a predefined number of times or until a stop condition is fulfilled: (a) selecting an initial sub-matrix, randomly or following a specific heuristic; and (b) performing an optimization process in which the initial sub-matrix is iteratively modified by using a state-or-the-art optimization algorithm or by adding and/or removing columns or rows, as long as a predefined function improves. Examples of algorithms following this approach are those reported in Cheng and Church (2000), Lazzeroni and Owen (2000) and Segal et al. (2003). The first of these three algorithms uses a function called *residue score* in step (b) whilst the algorithms reported in Lazzeroni and Owen (2000) and Segal et al. (2003) optimize, using a variation of the EM algorithm (Dempster et al. 1977), the sum of squared errors and a function based on a Gaussian distribution, respectively.

The second approach proposes to discover directly a predefined k number of biclusters. In order to accomplishing this goal, three ideas have been used.

The first idea consist in iteratively splitting a given matrix as long as a predefined function improves its vale. The algorithm reported in Hartigan (1972) follows this idea and it uses as function the overall variance of the matrix; posterior variations of this method were reported in Duffy and Quiroz (1991), Tibshirani et al. (1999), and they aim to explore the use of permutations in the splitting process and to induce the optimal number of biclusters, respectively.

The second idea is to apply a state-of-the-art clustering algorithm or an optimization algorithm over the columns (or rows) of the matrix in order to obtain a partition, and to use this partition in order to obtain a partition in the other dimension. CTWC (Getz et al. 2000), ITWC (Tang et al. 2001), DCC (Busygin et al. 2002) and the algorithm reported in (Robardet and Feschet 2001) follow this idea. CTWC uses a hierarchical clustering algorithm whilst ITWC and SCC use Kmeans and SOM, respectively; the algorithm proposed in Robardet and Feschet (2001) uses an optimization algorithm.

Finally, the third idea consists in starting with an initial set of biclusters which is iteratively modified by transforming (i.e., adding/removing objects or features) or merging some biclusters, until a predefined function is improved or a stop criterium is fulfilled. Examples of algorithms following this idea are FLOC (Yang et al. 2003), SAMBA (Tanay et al. 2002), the algorithms reported in (Califano et al. 2000; Liu and Wang 2003), and Maple (Pei et al. 2003), which was reported also in Wang and Pei (2008) as *pCluster* algorithm. FLOC uses the *residue score* function proposed in Cheng and Church (2000) whilst SAMBA models the matrix as a bipartite graph and it iteratively searches for the K heaviest bicliques in this graph. The algorithm reported in Liu and Wang (2003) follows the idea of SAMBA but it uses a modified prefix tree, named *OPC-tree*, and a variation of a sequential pattern mining algorithm for building the biclusters. The algorithm proposed in Califano et al. (2000) uses a pattern discovery method in order to find an initial set of biclusters that is then improved by a greedy heuristic. Finally, Maple starts by building an initial set of biclusters that contain

only two objects but they are maximal wrt. the features contained and biclusters that are described by only two features but they are maximal wrt. the objects they contain; the final set of biclusters is obtained through a merging heuristic.

The biclustering algorithms described in this section have a high computational complexity. Although the complexity of a biclustering algorithm depends on the exact problem formulation and the merit function used for evaluating the quality of the biclusters, almost all the interesting variants of this problem are NP-complete (Madeira and Oliveira 2004). Given this, many biclustering algorithms use heuristic approaches; nevertheless, they are still computational expensive. On the other hand, excepting the DCC (Busygin et al. 2002) and the algorithms proposed in Hartigan (1972), Robardet and Feschet (2001), all the other algorithms can build overlapping clusters. However, they can leave objects unclustered; that is, objects that do not belong to any bicluster.

Another limitation is that, excepting the algorithm proposed in Robardet and Feschet (2001), all the other algorithms need to tune up the values of two or more parameters used for building the biclusters, whose values depend on the data to process. Finally, most of these algorithms produce clusters whose concepts are difficult to understand in real problems, because they are very large.

2.9 Lattice-theory based algorithms

In this subsection, we review those algorithms which build a set of clusters as well as the concepts describing those clusters using concepts from Lattice Theory (Birkhoff 1967).

Examples of algorithms following this paradigm are MCC (Lee et al. 2007), GALOIS (Carpineto and Romano 1993), α-GALOIS (Ventos et al. 2004), OSHAM (Ho 1995) and INCOSHAM (Ho 1997), as well as the algorithms reported in Messai et al. (2008), Quan et al. (2004); Godin et al. (1995). These algorithms are based on *Formal Concept Analysis* (FCA), which is a data analysis method allowing to derive implicit relationships from a set of objects described by their attributes, using concepts of Lattice Theory (Birkhoff 1967).

Given a set O of elements (i.e., objects or set of objects), described in terms of a feature set D. A *formal concept*, hereinafter referred to as FC, is a pair (X, Y) such that $X \subseteq O$ is the set of objects which are unambiguously described by the feature set Y, and $Y \subseteq D$ is the set of features that are common to all the objects belonging to X; the set of objects of a FC constitutes the cluster whilst the set of features of this FC constitutes the description of those clusters.

MCC uses the EM algorithm (Dempster et al. 1977) and an exhaustive strategy for building a set of FC and then, it iteratively merges those FC highly similar; remaining FC are used for building a lattice using the subconcept and superconcept relationships existing among the FCs. GALOIS follows the idea of MCC but it processes the objects one by one, updating the lattice each time; α-GALOIS is an extension of GALOIS which aims to reduce the number of detected FCs. On the other hand, OSHAM assumes the FC comprised of all objects and features as the root of a hierarchy and it builds the remaining levels by iteratively partitioning the FC of each node using the subconcepts of that node which optimize a predefined function. INCOSHAM and the algorithm reported in Godin et al. (1995) follow the same strategy of OSHAM and GALOIS, respectively, but they are able to update the structure they build when new objects are added to the collection. Different from previous algorithms, the algorithm reported in Quan et al. (2004) extends the concepts of FCA so they take into account the fuzzy relations existing between objects and attributes; the fuzzy FCs obtained are afterwards processed using a fuzzy clustering algorithm for building a concept hierarchy. Finally, the

algorithm reported in Messai et al. (2008) follows the ideas of GALOIS but it is able to process objects described by multi-evaluated features.

Other two examples of conceptual algorithms following this paradigm are COING (Bournaud and Ganascia 1997) and KIDS (Bournaud et al. 2000). These algorithms assume that the objects are described by *conceptual graphs* (Chein and Mugnier 1992) and they build a lattice of clusters. COING starts by transforming the conceptual graph describing each object into a set of conceptual arcs and then, it clusters them, taking into account the objects these arcs describe; the resulting clusters are then used for building a lattice. KIDS also builds a lattice of clusters but it iteratively applies COING over the collection, using more complex structures for representing the objects each time.

The algorithms presented in this section have a high computational complexity. COING, KIDS, MCC, GALOIS and the algorithm proposed in Godin et al. (1995) can build a large number of clusters. Although α-GALOIS obtains fewer clusters than GALOIS, it needs the objects be previously labeled. MCC and the algorithm proposed in Quan et al. (2004) need to tune up the values of four and two parameters used for building the clusters (i.e., formal concepts) whose values depends on the collections to be processed. COING and KIDS build clusters whose concepts are difficult to understand since they are very large or the theory or the language used to build them is inherently difficult to understand. Finally, all these algorithms can build clusters with a high overlapping. Although there are contexts is which it is desirable to build overlapping clusters, when the overlapping among the clusters is too high, it could be difficult to obtain useful findings about the structure of the data.

A paradigm related with FCA is the Lattice Computing (LC) paradigm (Grana 2008). However, to the best of our knowledge the clustering algorithms developed under this paradigm do not explicit produce a set of clusters and their set of corresponding concepts or explanations, see (Kaburlasos and Petridis 2000; Kaburlasos et al. 2009, 2013). Moreover, most of the algorithms under this paradigm are supervised which is not the type of clustering algorithms we are analyzing in our work, see (Jamshidi and Kaburlasos 2014; Papadakis et al. 2014; Papakostas et al. 2015; Kaburlasos and Papakostas 2015; Esmi et al. 2016; Grana and Chyzhyk 2016). Based on the above two reasons we do not review those algorithms in this work.

2.10 Hybrid algorithms

In this subsection, we review those algorithms that combine more than one approach in order to build the set of clusters as well as their descriptions. Generally, the algorithms based on this approach perform the following steps: (a) building an initial set of clusters and/or concepts using a specific approach; and (b) processing the initial set using another approach in order to build the final conceptual clustering.

Examples of algorithms based on this approach are MFI Kmeans (Zhuang and Dai 2004), SHDC (Wang et al. 2007), ITERATE (Biswas et al. 1994), HGA-COBWEB (Yoo et al. 1996) and the algorithm reported in Hotho and Stumme (2002).

MFI Kmeans uses the MAFIA algorithm (Burdick et al. 2001) for computing the *maximal frequent itemsets* and consequently, the initial clusters; each itemset and its cover constitutes the concept and the cluster, respectively. These initial clusters are then processed in step (b) using the Kmeans. SDHC follows the same strategy as MFI Kmeans in step (b) but it uses in step (a) the parTFI algorithm (Yongheng et al. 2005) for obtaining the frequent patterns used for building the clusters. ITERATE uses a variation of the COBWEB algorithm (Fisher 1987) in order to build a hierarchy of cooncepts and then, it extracts from this hierarchy a partition that is processed in step (b) using an optimization approach and a function based on the

cohesiveness of clusters. HGA-COBWEB uses in step (a) a genetic algorithm for obtaining a set of clusters that are then processed in step (b) using a variation of COBWEB/3 (McKusick and Thompson 1990) for building a hierarchy of concepts. Finally, the algorithm proposed in Hotho and Stumme (2002) uses the Bisecting Kmeans algorithm (Zhao and Karypis 2002) for obtaining the initial clusters that are then processed using Formal Concept Analysis in order to build a lattice of concepts.

Another two hybrids algorithms are LINNEO+ (Béjar and Cortés 1992) and MC (Zhang et al. 2010). LINNEO+ (Béjar and Cortés 1992) is an incremental conceptual algorithm that builds an initial set of clusters following a simple-pass approach. Aterwards, some of the processed objects are reprocessed using a local optimization process and a predefining function based on cluster cohesiveness; LINNEO+ describes a cluster using a set of attribute-value pairs built from the objects included in the clusters. On the other hand, MC represents the documents of a collection using the frequent itemsets extracted by the Apriori algorithm (Agrawal et al. 1993). Starting from this point, MC iteratively selects the more similar pair of objects and, depending on the membership of these objects to the existing clusters, MC creates a new cluster, or it merges two existing clusters, or it includes an unclustered object into an already created cluster. The concept describing each cluster is built using the frequent itemsets with highest support inside the cluster.

The algorithms described in this section, excepting ITERATE and MC, need to tune up the value of several parameters using for building the clusters, whose values depend on the collections to be processed. It is important to highlight that although MC only depends on one parameter (i.e., the minimum support used for computing the frequent itemsets), this parameter has a high impact in both the efficiency and the accuracy of MC. The MC algorithm requieres a lot of computer memory whilst ITERATE, HGA-COBWEB and the algorithm proposed in Hotho and Stumme (2002) have a high computational complexity. Besides, ITERATE, HGA-COBWEB and LINNEO+ build clusters whose concepts are difficult to understand since they are very large or the theory or the language used to build them is inherently difficult to understand. Finally, although the redistribution process ITERATE performs focused on to reduce the data order dependency, ITERATE still depends on data order.

3 Qualitative comparison of the algorithms

Comparing the algorithms described in the previous sections in terms of some evaluation measure could be not fair, mainly because most of them were not designed for processing the same kind of objects, and also because they are different in terms of the structure in which they organize the collection of objects, among other factors. However, we can compare them qualitatively, attending to some important characteristics that we think should be present in a conceptual clustering algorithm.

The idea of this comparison is not to determine which is the best algorithm, because in clustering this is an ambiguous term, which depends on each particular problem, what the users expect in the results, as well as on the way in which the output of the algorithms is validated. The main goal of this comparison is to illustrate the trade-of each algorithm offers with respect to the proposed characteristics and thus, to help the selection of an appropriate algorithm to solve a problem at hand.

Following, we first present which are the characteristics that will be used in our qualitative comparison of the algorithms, and also we argue about the importance of these characteristics.

Finally, a comparison of the algorithms in terms of the above mentioned characteristics is given and some important findings are discussed.

3.1 Characteristics to take into account in the comparison

A characteristic that should be taken into account when an algorithm is analyzed is its computational complexity, even more if this algorithm will be used in real problems. The computational complexity of algorithms is generally expressed assuming the worst scenario the algorithms could address and thus, the lower this complexity is the highest will be the amount of data the algorithms will be able to process.

In the 1990s, the algorithms having a $O(n^2)$ complexity were considered efficient taking into the account the size of the datasets commonly processed in that time. Nowadays, there are applications dealing with datasets having a size of several terabytes or even petabytes; algorithms with a quadratic computational complexity could be inefficient or even not useful since they could be unable to process such amount of data in a reasonable short period of time. Based on the above analysis, the computational complexity of the algorithms will be taken into account in the comparison. This variable will be referred to as "Compl." and it will take the following values:

- *v. high*, for those algorithms whose complexity is $O(n^5)$ or greater.
- *high*, for algorithms having a complexity of $O(n^3)$ or greater.
- *middle*, if the complexity of the algorithm is $O(n^2)$ or greater.
- *low*, for those algorithms having a complexity lower than $O(n^2)$.

Another important characteristic to take into account is the number of parameters of the algorithm; the highest this number the more difficult will be to tune up them and thus, the more difficult will be to use the algorithm, specially in a real problem. Moreover, there are parameters that have a high impact into the efficiency and accuracy of the algorithms, making it even more difficult to find a proper value for them. Based on these reasons, the number of parameters an algorithm has will be taken into account in our comparison; this variable will be referred to as "N. par.".

Another characteristic that is also useful in our comparison, is the type of data structure the algorithm builds to store the concepts and the clusters. Generally, most of the conceptual clustering algorithms reported in the literature have been focused on building a set of clusters or a hierarchy; however; there have been algorithms able to produce a lattice of clusters or a conceptual graph. Based on the aforementioned analysis, the type of data structure an algorithm builds for storing the clustering result will be taken into account in our comparison, and it will be referred to as "Struct.". This variable will take the values *set*, *hier.*, *latt.* or *graph*, depending on the structure the algorithm builds.

The objects analyzed on most of real problems are generally described by quantitative and qualitative features. Moreover, there are problems where the values of some features describing several objects are missing. Although there are algorithms reported in the literature that work with objects described by mixed data, when they are going to compare objects most of them use one of the following approaches: (a) they compare the objects considering first the quantitative features and second the qualitative ones and then, they do a conjoint answer, or (b) they transform the numeric features into qualitative ones or vice versa, in this way all the features are handled as if they were of the same type. In the case of incomplete or missing data, most of the algorithms perform one of the following steps: (a) they estimate values for the missing data, or (b) they do not consider as part of the description of the objects the features having missing values. Thus, the type of features an algorithm is able to process

will be also taken into account in our comparison, and it will be referred to as "T. att.". This variable will take the values *num*, *qual*, *mixed*, or *mix-mis*, depending on the type of features the algorithm is able to process.

Another characteristic that will be used in our comparison, is the kind of cluster the algorithm builds. The clusters built by an algorithm can be disjoint, fuzzy or they can be overlapping. Most of the clustering algorithms reported in the literature have been focused on to build disjoint clusters. However, lately there have been proposed several algorithms for building overlapping clusters and some other which produce fuzzy clusters. Thus, the type of clusters an algorithm builds will be taken into account in our comparison and it will be referred to as "T. clus.". This variable will take the values *disj*, *overl* or *fuzzy*, depending on the clusters built by the algorithm.

There are problems in which the whole data changes due to additions, deletions or even modifications of the objects. How an algorithm processes these changes in order to update the clustering is another characteristic that we will use in our comparison. According to this characteristic, the algorithms can be *static*, *incremental* or *dynamic*. The static algorithms are those that each time the collection changes, they need to reprocess the whole collection from scratch in order to update the clusters. Incremental algorithms are able to use the clusters built so far in order to update the current clustering when objects are added to the collection. Finally, dynamic algorithms are those whose are able to efficiently update the current clustering after additions, deletions and even modifications of objects, using the clusters built so far. In our comparison, this characteristic of the algorithms about the capability to efficiently deal with changes in an already clustered collection will be taken into account and it will be referred to as "Type". This variable will take the values *static*, *incr* or *dyna*, depending on the nature of the algorithm.

Finally, a primary characteristic that should be taken into account for comparing conceptual algorithms is how difficult to understand are the concepts these algorithms generate for describing the clusters they build. Some times, the algorithms generate concepts based on probability theory, conceptual graphs or other description language which might result difficult to understand for final users that are not specialists is those fields. Other times, the language in which the concepts are described is easy to understand but the size of the generated concept makes difficult its understanding. Therefore, the easiness of a generated concept to be understood by an average user will be taken into account in our comparison, and it will be referred to as "C. int.". This variable will take the values *hard*, *med* and *easy*, depending of how difficult the understanding of the concepts generated by an algorithm might be for an average final user.

3.2 Comparison of the algorithms and discussion

It is important to mention that in the comparisons presented in this section the following algorithms were not taken into account since they are slightly versions of other algorithms that already appear in our comparison:

- CLUSTER/PAF and CLUSTER/S; the first one could be viewed as an obsolete version of CLUSTER/2, while the strategy followed by the second one is basically the application of CLUSTER/2.
- AGAPE does not differ to much from the DF algorithm.
- CLASSIT+ follows exactly the same strategy as CLASSIT.
- The algorithm proposed in Anaya-Sánchez et al. (2010) follows the same general strategy as the algorithm proposed in Anaya-Sánchez et al. (2008).
- FTSC follows practically the same strategy used by FTC.

- FCDC and the algorithm proposed in Kryszkiewicz and Skonieczny (2006) are extensions of the FIHC algorithm.
- CFWMS employs basically the same strategy as CFWS.
- PCN employs basically the same strategy as PCUDT.
- The algorithms proposed in Duffy and Quiroz (1991), Tibshirani et al. (1999) are simple variations of the method proposed by Hartigan in Hartigan (1972).
- ITWC and DCC, follow the same general strategy of CTWC (Getz et al. 2000).
- α-GALOIS and the algorithms proposed in Godin et al. (1995),Messai et al. (2008), employ the same general strategy of GALOIS (Carpineto and Romano 1993).

Tables 1 and 2 show the qualitative comparison among the conceptual clustering algorithms described in the previous section, according to the selected characteristics. In both of these tables there are several values appearing with a superscript. Following we explain which is the meaning of each superscript appearing on each column:

- column "Compl.". In this column, the algorithms FTC, HFTC, FTSHC, FIHC, TDC, GPHC, IDHC, MC, FTMTC, MFI Kmeans and SHDC, as well as the algorithm proposed in Krishna and Bhavani (2010) appear with the superscript "a", for indicating that although the computational complexity of these algorithms is generally not so high, this complexity depends on the value of the support threshold used for computing the frequent patterns the algorithms use for clustering. As it was mentioned before, low values of this parameter could make the algorithms computational expensive or even intractable, while high values could make the algorithms to produce clusters of low quality due to the missing of interesting frequent patterns.
- column "T. att.". In this column, the algorithms CKM and ITERATE appear with a superscript "b", for indicating that although they can handle objects described by both qualitative and quantitative features, they achieve that goal by transforming qualitative features into numeric ones or vice versa. The algorithm LINNEO+ appears with a superscript "c", for indicating that although it can handle objects whose description is incomplete (i.e., missing data), it achieves this goal by not taking into account those features or by assigning them a value computed from the data.
- column "T. clus.". In this column, the algorithms WITT, GPHC, CPC, CTWC, FLOC, SAMBA, EMO-CC and the algorithms proposed in Cheng and Church (2000), Califano et al. (2000), Liu and Wang (2003), Lazzeroni and Owen (2000), Segal et al. (2003) appear with a superscript 'd", for indicating that they can leave unclustered objects. Besides, the algorithm FTC appears with a superscript "e", for indicating that although it builds disjoint clusters, the authors of FTC propose a simple modification of its control structure in order to make it able to build overlapping clusters.
- column "N. par.". In this column, the algorithm proposed in (Anaya-Sánchez et al. 2008) appears with a superscript "f", for indicating that there is a version of this algorithm, published in Anaya-Sánchez et al. (2010), that includes a strategy for automatically computing its parameter value.
- column "C. int.". In this column, the algorithms EPAM, HDCC and LINNEO+ appear with a superscript "g", for indicating that the concepts these algorithms build for describing the clusters might not cover all the objects in these clusters; that is, it could happen that one or more objects in a cluster do not match the concept describing that cluster. Besides, the algorithm proposed in Anaya-Sánchez et al. (2008) appears with a superscript "h" meaning that, although the concepts this algorithm builds are easy to understand, their authors propose a variant in Anaya-Sánchez et al. (2010) that builds concepts which could be more difficult to understand.

Based on the information showed in Tables 1 and 2, it can be seen that more than one half of the conceptual clustering algorithms reported in the literature have a high or very high computational complexity, so in real problems dealing with a large number of objects they are useless. On the other hand, it is important to highlight that many of the algorithms having a middle computational complexity are interesting-pattern based. These algorithms depend on the value of the support threshold used for building the frequent patterns; low values of this parameter could make these algorithms computational expensive or even intractable, while high values could make the algorithms to produce clusters of low quality due to the missing of interesting frequent patterns. Thus, in practice, these algorithms could also be computational expensive. Moreover, there is only one algorithm having a low computational complexity: the W-Kmeans algorithm, which was specifically designed for processing documents (Bouras and Tsogkas 2010). W-kmeans is based on Kmeans and it uses WordNet for enriching the description of the documents as well as in the process of building the concepts of the clusters. W-kmeans depends only on one parameter and the concepts it builds for explaining the clusters are easy to understand. However, W-kmeans is a static algorithm so it can be not useful in applications processing collections that can change over the time. Finally, W-kmeans is only able to process objects described by quantitative features and it needs to know in advance the number of clusters to build. Generally, users do not have any a priori knowledge about the data they want to cluster and this is why it could be difficult in real problems to set up a proper number of clusters before-hand.

Most of the conceptual clustering algorithms have been focused on building sets or hierarchies of clusters, being those producing sets of clusters the most studied. On the other hand, there are five algorithms that organize the collection into a lattice of clusters and one algorithm which builds a conceptual graph; however, all of them have a high computational complexity.

Among the algorithms organizing the collection into a set of clusters, FTC, W-kmeans and the algorithm proposed in Anaya-Sánchez et al. (2008) are the ones offering the best trade off in terms of the evaluated characteristics. The limitations of W-kmeans were previously stated. On the other hand, both FTC and the algorithm proposed in Anaya-Sánchez et al. (2008) are static algorithms and they are only able to process objects described by qualitative features. Moreover, FTC is a interesting-pattern based algorithm that depends on the support threshold used for computing the frequent patterns it uses; as it was mentioned before, its computational complexity could be very high depending on the value of the this threshold. On the other hand, the hierarchical clustering algorithms having the best trade-off are HFTC, FTSHC and FIHC. Like FTC, these three hierarchical algorithms are interesting-pattern based and they are static algorithms. Finally, GALOIS is the best algorithm among the algorithms building a lattice; however, as it was previously stated it has a high computational complexity and the concepts it builds could be difficult to understand.

It is worth of mentioning that, as it can be seen in Tables 1 and 2, most of the existing conceptual clustering algorithms are static algorithms; that is, when an already clustered collection changes, they need to reprocess the whole new collection in order to update the current clustering from scratch. On the other hand, excepting COBBIT, INC, WebDCC and LINNEO+ algorithms, the remaining conceptual clustering algorithms able to deal with changes have a high computational complexity; thus, they could be not suitable for processing large collections in real problems. Although COBBIT, INC, WebDCC and LINNEO+ are less computational expensive than the other algorithms, they build concepts that could be difficult to understand in real problems, either because they are very large or because the theory or the description language used to build them is inherently difficult to understand. In case that the computational complexity were not a matter and we were looking for conceptual algorithms

Table 1 Qualitative comparison among algorithms, attending to the selected characteristics

Alg.	Compl.	Struct.	Type	T. att.	T. clus.	N. par.	C. int.
CLUSTER/2	v. high	set	static	mixed	disj.	4	med.
CLUSTER/3	v. high	hier.	static	mixed	disj.	4	med.
CKM	middle	set	static	mixed[b]	disj.	5	med.
CKMSF	middle	set	static	mixed	disj.	3	med.
CKMCF	high	set	static	mixed	disj.	3	med.
DF	high	set	static	qual.	disj.	3	med.
W-Kmeans	low	set	static	num.	disj.	1	easy
CoReClus	middle	set	static	qual.	disj.	4	hard
COBWEB	high	hier.	inc.	qual.	disj.	0	hard
COBWEB/3	high	hier.	inc.	mixed	disj.	0	hard
CLASSIT	high	hier.	inc.	num.	disj.	0	hard
DynamicWeb	v. high	hier.	dyn.	qual.	disj.	1	hard
ARACHNE	v. high	hier.	inc.	qual.	disj.	0	hard
COBBIT	middle	hier.	dyn.	qual.	disj.	1	hard
BRIDGER	v. high	hier.	inc.	mixed	disj.	0	hard
INC	middle	hier.	dyn.	mixed	disj.	3	hard
LABYRINTH	v. high	hier.	inc.	qual.	disj.	0	hard
WebDCC	middle	hier.	inc.	num.	disj.	4	med.
Perner and Attig (2010)	high	hier.	inc.	num.	fuzzy	3	med.
WITT	middle	set	static	qual.	disj.[d]	3	hard
HDCC	middle	hier.	static	num.	disj.	0	med.[g]
GCC	middle	hier.	static	qual.	disj.	2	hard
OPUS	v. high	hier.	static	qual.	disj.	0	hard
M-DISC	high	hier.	static	num.	disj.	1	hard
LC	v. high	set	static	mix-mis	disj.	1	med.
RGC	v. high	set	static	mix-mis	disj.	1	med.
Anaya-Sánchez et al. (2008)	middle	set	static	num.	disj.	1[f]	easy[h]
Pons-Porrata et al. (2007)	v. high	hier.	inc.	num.	disj.	1	hard
FTC	middle[a]	set	static	qual.	disj.[e]	1	easy
HFTC	middle[a]	hier.	static	qual.	overl.	1	easy
FTSHC	middle[a]	hier.	static	qual.	overl.	1	easy
FIHC	middle[a]	hier.	static	qual.	disj.	1	easy
TDC	middle[a]	hier.	static	qual.	overl.	2	easy
GPHC	middle[a]	hier.	static	qual.	overl.[d]	3	easy
Krishna and Bhavani (2010)	middle[a]	set	static	qual.	disj.	3	easy
F^2IHC	high	hier.	static	qual.	disj.	6	easy
IDHC	middle[a]	hier.	static	qual.	overl.	5	easy
Kiran et al. (2010)	high	hier.	static	qual.	overl.	4	hard
CFWS	high	set	static	qual.	overl.	2	hard
MC	middle[a]	set	static	qual.	disj.	1	med.

Table 2 Qualitative comparison among algorithms, attending to the selected characteristics (cont.)

Alg.	Compl.	Struct.	Type	T. att.	T. clus.	N. par.	C. int.
CPC	high	set	static	qual.	disj.[d]	5	easy
PCUDT	middle	set	static	qual.	disj.	5	easy
FTMTC	middle[a]	set	static	qual.	overl.	2	easy
Hartigan (1972)	v. high	set	static	num.	overl.	2	hard
CTWC	high	set	static	num.	overl.[d]	2	hard
Cheng and Church (2000)	v. high	set	static	num.	overl.[d]	3	hard
FLOC	v. high	set	static	num.	overl.[d]	2	hard
Califano et al. (2000)	v. high	set	static	num.	overl.[d]	1	hard
Maple	high	set	static	num.	overl.	3	hard
Robardet and Feschet (2001)	high	set	static	qual.	disj.	0	hard
SAMBA	v. high	set	static	num.	overl.[d]	5	hard
Liu and Wang (2003)	v. high	set	static	num.	overl.[d]	2	hard
Lazzeroni and Owen (2000)	high	set	static	num.	overl.[d]	2	hard
Segal et al. (2003)	high	set	static	num.	overl.[d]	1	hard
EMO-CC	middle	set	static	qual.	disj.[d]	3	hard
Fanizzi et al. (2007)	middle	set	static	num.	disj.	3	hard
GAIL	middle	set	static	qual.	disj.	2	easy
MCC	high	graph	static	num.	overl.	4	med.
GALOIS	high	latt.	inc.	qual.	overl.	0	med.
Quan et al. (2004)	v. high	latt.	static	num.	fuzzy	2	med.
OSHAM	v. high	latt.	static	qual.	overl.	0	med.
INCOSHAM	v. high	latt.	inc.	qual.	overl.	0	med.
Hotho and Stumme (2002)	high	latt.	static	num.	overl.	2	med.
MFI Kmeans	middle[a]	set	static	num.	disj.	3	easy
SHDC	middle[a]	set	static	num.	disj.	3	easy
ITERATE	v. high	set	static	mixed[b]	disj.	0	hard
HGA-COBWEB	v. high	hier.	inc.	mixed	disj.	3	hard
EPAM	high	hier.	inc.	qual.	disj.	0	med.[g]
COING	high	set	static	qual.	disj.	0	hard
KIDS	v. high	set	static	qual.	disj.	0	hard
UNIMEN	high	hier.	inc.	mixed	overl.	5	easy
RESEARCHER	high	hier.	inc.	mixed	overl.	5	easy
LINNEO+	middle	set	inc.	mix-mis[c]	disj.	1	hard[g]

able to both deal with changes and to produce concepts easy to understand, our best choice were UNIMEN and RESEARCHER; however, they require to tune up five parameters using for building the clusters, which could depend on the collection to process. In general, users do not have any a priory knowledge about the collection they want to cluster; therefore, to tune up several parameters could be a difficult task.

Only fifteen algorithms are able to process objects described by mixed features and most of them have a high computational complexity, being the best ones CKM, CKMSF, INC and

LINNEO+; however, excepting this last one, the other three algorithms require to tune up three or more parameters, which is not easy in real problems. Moreover, among the algorithms that can deal with mixed data, only three of them can deal with missing data: RGC, LC and LINNEO+; however, these algorithms have a high computational complexity or they build concepts that can be difficult to understand in real applications. It is important to take into account that although the description of the objects of a real problem can be transformed in order to be conformed only by qualitative or numeric features, this transformation can have a negative impact in the accuracy of the results since the inherent semantics of the problem is changed.

As it was expected most of the algorithms build disjoint clusters and excepting some cases, almost all the conceptual clustering algorithms able to build overlapping clusters are interesting-pattern based, or they follows the biclustering or the Lattice-Theory paradigms. At the same time, most of the overlapping conceptual clustering algorithms have a high computational complexity, they depend on several parameters or they build concepts which can be difficult to understand in real problems. The conceptual overlapping clustering algorithms having the best trade off in terms of the evaluated characteristics are FTC, HFTC and FTSHC, whose limitations have been previously stated. On the other hand, it is interesting to find out that only two algorithms are able to build fuzzy clusters but they are highly computational expensive.

Finally, more than one half of the algorithms depend on two or more parameters, while most of the algorithms depending on just one or zero parameters have a high computational complexity or they build concepts difficult to understand. It is also interesting to observe that most of the conceptual clustering algorithms build concepts that are difficult to understand since these concepts are very large or they are expressed in a language that could be difficult to understand for an average final user. Among the algorithms producing descriptions which can be easy to understand the best ones are FTC, HFTC, FTSHC, FIHC, W-kmeans and the algorithm reported in Anaya-Sánchez et al. (2008), being the fist four based on frequent patterns.

For summarizing, we show in Table 3 showed the main positive and negative characteristics of each paradigm reviewed in this work.

4 Application of conceptual clustering algorithms

The conceptual clustering algorithms developed in several works have been applied in different real-life problems such as Video Surveillance, Bioinformatic, Information Retrieval and Browsing, and Cibersecurity, among others.

The works (Bellotto et al. 2012; Lee et al. 2007) show how conceptual clustering can be used for video surveillance tasks. In particular, in Lee et al. (2007) the MCC conceptual algorithm is used for grouping a set of trajectories in order to perform moving-objects tracking whilst in Bellotto et al. (2012) a conceptual clustering algorithm is used for cognitive visual tracking and for improving the camera control.

Besides, several conceptual algorithms based on the biclustering paradigm have been used in bioinformatic for gene expression data analysis. One of the goals of this analysis is to cluster genes according to their expression under multiple conditions or clustering conditions based on the expression of a number of genes. Discovering these clusters may lead for uncovering many genetic pathways that are not apparent and for example, it may help in understanding several diseases and consequently, in the developing of effective treatments (Madeira and

Table 3 Main positive and negative characteristics of each paradigm

Paradigm	Positive	Negative
Optimization based	Most of the algorithms have a high complexity	Most of them build concepts easy to understand
	They depend on a lot of parameters	Many of them process mixed data
	They are static–they only build disjoint clusters	
Tree based	Most of the algorithms have a very high complexity	They depend on few parameters–they process changes in the collection
	They build concepts difficult to understand	
	Most of them build disjoint cluster	
Agglomerative	They are static	They have a relative low complexity
	They depend on a lot of parameters	
	They only build disjoint clusters	
Divisive	They have a high complexity	They depend on few parameters
	They build concepts difficult to understand	
	They are static	
	They only build disjoint clusters	
Graph based	They have a high complexity	They depend on few parameters
	They are static	They process mixed data
Interesting-Patterm based	They are static	Most of them build concepts easy to understand
	None of them process mixed data	
		They have a relative low complexity
		Many of them build overlapping clusters
Biclustering	They have a high complexity	They build overlapping clusters
	They build concepts difficult to understand	
	They are static	
Evolutional based	They only build disjoint clusters	They have a relative low complexity
	They build concepts difficult to understand	
	They are static	
Lattice-Theory based	They have a high complexity	They build overlapping an even fuzzy clusters
		Most of them depend on few parameters
Hybrid	They only build disjoint clusters	Some of them process mixed data
	They have a relative high complexity	
	They are static	

Oliveira 2004). Examples of biclustering algorithms used for this purpose are reported in Tang et al. (2001), Busygin et al. (2002),Tanay et al. (2002),Liu and Wang (2003). Additionally, the work in Hartigan (1972) uses a biclustering algorithm over electoral data in order to discovering countries with the same political ideas and electoral behaviors.

With the developing of network technology, the popularity of network and cloud applications have increase; however, the number and severity of attacks over those applications have also increase (Bhuyan et al. 2017). Examples on conceptual algorithms developed for Cibersecurity were reported in Xu (2014), Mulani et al. (2015) for intrusion detection and privacy preservation in cloud database querying, respectively.

Conceptual clustering techniques haven been used also for enhance the efficiency and effectiveness of Information Retrieval Systems and for Web Browsing (Bhatia and Deogun 1998). Examples of conceptual algorithm used for this purpose are reported in Malik et al. (2010), Wang and Liu (2011), Zheng et al. (2014), Spanakis et al. (2012), DAmato et al. (2010).

In addition to the above mentioned applications, conceptual clustering algorithm have been also applied for news and documents clustering, see (Pons-Porrata et al. 2007; Bouras and Tsogkas 2010; Krishna and Bhavani 2010); for Real-Time Optimal Selection of Multirobot Coalition Formation Algorithms see (Sen and Adams 2015); and for web user profiling, see (Godoy and Amandí 2006).

5 Concluding remarks

The possibility of building a set of clusters as well as the concepts that describe those clusters is a desirable property of any clustering algorithm.

In this paper, we present an overview of the most influential conceptual clustering algorithms reported in the literature, highlighting their limitations. Additionally, we have proposed a taxonomy of the reviewed algorithms and we have conducted a qualitative comparison among the analyzed algorithms, in terms of a set of characteristics that are desirable in any conceptual clustering algorithm. It is our hope that this comparison will help the researchers in the selection of an appropriate algorithm to solve a problem at hand.

From the analysis of the different algorithms and paradigms, we consider that in future works the following issues should be tackled.

First of all, it is necessary to development of conceptual clustering algorithms able to efficiently process additions, eliminations as well as modifications of objects. It is known that nowadays many applications process very large collections of objects that can suffer changes due to additions, eliminations or modifications of objects. Most of the conceptual clustering algorithms that can efficiently process changes in an already clustered collection have been focused only on processing additions. Only, COBBIT (Kilander and Jansson 1993) and DynamicWeb (Scanlan et al. 2008) algorithms are able to process elimination of objects but they are highly computational expensive and they build concepts that are difficult to understand. These two characteristics make them less useful in real problems.

In order to increase the application of conceptual clustering algorithms in real problems it is necessary the development of efficient and accurate conceptual clustering algorithms able to process objects described by both mixed and missing features. Most of the conceptual clustering algorithms reported so far are not able to process objects described such kind of features; however, in a real problem it is usual to work with these objects. Most of the reported approaches that deal with mixed and/or incomplete features transform the features

describing the objects or they ignore those parts of the description of the objects that might be problematic.

Many real applications nowadays are taking into account the possibility that an object could belong to one or more clusters, possibly with different membership degrees. Combining the building of fuzzy clusters and the generation of concepts that can accurately describe those clusters is a challenge; therefore, we think that further research in conceptual algorithms must focused on to produce overlapping and even fuzzy clusters. As it was observed in this work, the study of conceptual clustering algorithms have been mainly focused on producing disjoint clusters and most of the approaches that are able to build overlapping or fuzzy clusters are highly computational expensive or they have other characteristics that reduce their application in real problems.

Taking into account that the interesting-pattern based algorithms are those having the best trade off in term of the evaluated characteristics and knowing that all of these algorithms are static, we consider that further developments under this paradigm must address the efficient updating of the clusters and concepts after additions as well as deletions of objects.

Finally, taking into account that the size of the collections processed in several applications grows daily, like for instance those dealing with malicious activity detection (Herrera-Semenets et al. 2017), we consider that another important and open venue in the field of conceptual clustering is the development of parallel or distributed conceptual clustering algorithms able to process very large collections; i.e., big data.

References

Agrawal R, Imielinski T, Swami AN (1993) Mining association rules between sets of items in large databases. In: Proceedings of the ACM SIGMOD international conference on management of data (SIGMOD93), pp 207–216

Anaya-Sánchez H, Pons-Porrata A, Berlanga-Llavorí R (2008) A new document clustering algorithm for topic discovering and labeling. In: Proceedings of the 12th iberoamerican congress on pattern recognition (CIARP 2008), LNCS 5197, pp 161–168

Anaya-Sánchez H, Pons-Porrata A, Berlanga-Llavorí R (2010) A document clustering algorithm for discovering and describing topics. Pattern Recogn Lett 31(6):502–510

Ayaquica-Martínez IO, Martínez-Trinidad JF, Carrasco-Ochoa JA (2005) Conceptual K-means algorithm with similarity functions. In: Proceedings of the 9th Iberoamerican congress on pattern recognition (CIARP 2005), LNCS 3773, pp 368–376

Ayaquica-Martínez IO, Martínez-Trinidad JF, Carrasco-Ochoa JA (2006) Conceptual K-means algorithm based on complex features. In: Proceedings of the 10th Iberoamerican congress on pattern recognition (CIARP 2006), LNCS 4225, pp 491–501

Baghel R, Dhir R (2010) A frequent concepts based document clustering algorithm. Int J Comput Appl 4(5):6–12

Beil F, Ester M, Xu X (2002) Frequent term-based text clustering. Proc FGML Workshop 37:436–442

Béjar J, Cortés U (1992) LINNEO+: Herramienta Para la Adquisición de Conocimiento y Generación de Reglas de Clasificación en Dominios Poco Estructurados. In: Proceedings of the III Congreso Iberoamericano de Inteligencia Artificial (IBERAMIA 92), La Habana, Cuba, pp 471–781

Bellotto N, Benfold B, Harland H, Nagel H, Pirlo N, Reid I, Sommerlade E, Zhao C (2012) Cognitive visual tracking and camera control. Comput Vis Image Underst 116(3):457–471

Bhatia SK, Deogun JS (1998) Conceptual clustering in information retrieval. IEEE Trans Syst, Man, Cybern, Part B (Cybern) 28(3):427–436

Bhuyan HM, Bhattacharyya D, Kalita JK (2017) Network traffic anomaly detection and prevention: concepts, techniques, and tools. Springer International Publishing, chap Network Traffic Anomaly Detection Techniques and Systems, pp 115–169

Birkhoff G (1967) Lattice theory. American Mathematical Society, Providence

Biswas G, Weinberg J, Li C (1994) ITERATE: a conceptual clustering method for knowledge discovery in databases. In: Proceedings of innovative applications of artificial intelligence in the oil and gas industry, pp 111–139

Bouras C, Tsogkas V (2010) W-kmeans: clustering news articles using wordnet. In: Proceedings of the 14th international conference on Knowledge-based and intelligent information and engineering systems: Part III, pp 379–388

Bournaud I, Ganascia JG (1997) Accounting for domain knowledge in the construction of a generalization space. In: Proceedings of the third international conference on conceptual structures (ICCS97), pp 446–459

Bournaud I, Courtine M, Zucker JD (2000) Abstractions for knowledge organization of relational descriptions. In: Proceedings of the 4th international symposium on abstraction, reformulation and approximation (SARA2000), pp 87–106

Burdick D, Calimlim M, Gehrke J (2001) Mafia: A Maximal frequent itemset algorithm for transactional database. In: Proceedings of the 17th international conference on data engineering, pp 443–452

Busygin S, Jacobsen G, Kramer E (2002) Double conjugated clustering applied to leukemia microarray data. In: Proceedings of the second SIAM international conference on data mining, workshop clustering high dimensional data

Califano A, Stolovitzky G, Tu Y (2000) Analysis of gene expression microarays for phenotype classification. In: Proceedings of the international computational molecular biology, pp 75–85

Carpineto C, Romano G (1993) Galois: an order-theoretic approach to conceptual clustering. In: Proceedings of the tenth international conference on machine learning, pp 33–40

Chein M, Mugnier ML (1992) Conceptual Graphs: fundamental notions. Revue d'Intelligence Artificielle 6(4):365–406

Cheng Y, Church GM (2000) Biclustering of expression data. In: Proceedings of the eighth international conference on intelligent systems for molecular biology, pp 93–103

Chu WW, Chiang K, Hsu C, Yau H (1996) An error-based conceptual clustering method for providing approximate query answers. Commun ACM 39(12):216

Chun-Ling C, Tseng FS, Liang T (2010) Mining fuzzy frequent itemsets for hierarchical document clustering. Inf Process Manage 46(2):193–211

DAmato C, Staab S S, Fanizzi N, Exposito F (2010) Dl-link: a conceptual clustering algorithm for indexing description logics knowledge bases. Int J Semant Comput 4(4):453–486

Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197

Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the EM algorithm. J Roy Stat Soc: Ser B (Methodol) 39(1):1–38

Duffy D, Quiroz A (1991) A permutation based algorithm for block clustering. J Classif 8:65–91

Esmi E, Sussner P, Sandri S (2016) Tunable equivalence fuzzy associative memories. Fuzzy Sets Syst 292:242–260

Fanizzi N, Amato C, Esposito F (2007) Evolutionary conceptual clustering of semantically annotated resources. In: Proceedings of the international conference on semantic computing (ICSC2007), pp 783–790

Feigenbaum EA (1961) The simulation of verbal learning behavior. In: Proceedings of the western joint IRE-AIEE-ACM computer conference, pp 121–132

Fisher DH (1987) Knowledge acquisition via incremental conceptual clustering. Mach Learn 2(2):139–172

Fonseca NA, Santos-Costa V, Camacho R (2012) Conceptual clustering of multi-relational data. Proc ILP 2011:145–159

Fore N, Dong G (2012) Contrast data mining: concepts, algorithms and applications. Chapman and Hall/CRC, London, chap CPC: A Contrast Pattern Based Clustering Algorithm, pp 197–216

Funes A, Ferri C, Hernández-Orallo J, Ramírez-Quintana MJ (2008) Hierarchical distance-based conceptual clustering. In: Proceedings of ECML PKDD 2008, LNAI 5212, pp 349–364

Fung B, Wangand K, Ester M (2003) Hierarchical document clustering using frequent itemsets. In: Proceedings of the SIAM international conference on data mining 2003

García-Serrano JR, Martínez-Trinidad JF (1999) Extension to K-means algorithm for the use of similarity functions. In: Proceedings of the third european conference on principles of data mining and knowledge discovery proceedings. Prague, Czech. Republic, pp 354–359

Gennari JH, Langley P, Fisher DH (1989) Models of incremental concept formation. Artif Intell 40(1–3):11–61

Getz G, Levine E, Domany E (2000) Coupled two-way clustering analysis of gene microarray data. In: Proceedings of the national academy of sciences of the United States of America, pp 12,079–12,084

Godin R, Missaoui R, Alaoui H (1995) An approach to concept formation based on formal concept analysis. Comput Intell 11(2):246–267

Godoy D, Amandí A (2006) A conceptual clustering approach for user profiling in personal information agents. AI Commun 19(3):207–227
Grana M (2008) Lattice computing: lattice theory based and computational intelligence. In: Proceedings of Kosen workshop on mathematics, technology and education
Grana M, Chyzhyk D (2016) Image understanding applications of lattice autoassociative memories. IEEE Trans Neural Netw Learn Syst 27(9):1920–1932
Gutiérrez-Rodríguez AE, Martínez Trinidad JF, García-Borroto M, Carrasco-Ochoa JA (2015a) Mining patterns for clustering on numerical datasets using unsupervised decision trees. Knowl-Based Syst 82:70–79
Gutiérrez-Rodríguez AE, Martínez Trinidad JF, García-Borroto M, Carrasco-Ochoa JA (2015b) Mining patterns for clustering using unsupervised decision trees. Intell Data Anal 19(6):1297–1310
Hadzikadic M, Yun DYY (1989) Concept formation by incremental conceptual clustering. In: Proceedings of the international joint conference artificial intelligence, pp 831–836
Han J, Pei J, Yin Y, Mao R (2004) Mining frequent patterns without candidate generation: a frequent-ppattern tree approach. Data Min Knowl Disc 8(1):53–87
Han MM, Tatsumi S, Okumoto T (1997) Applying genetic algorithm to conceptual clustering. IEEJ Trans Electron, Inf Syst 117(C(8)):1140–1151
Hanson SJ, Bauer M (1989) Conceptual clustering, categorization, and polymorphy. Mach Learn 3(4):343–372
Hartigan J (1972) Direct clustering of a data matrix. J Am Stat Assoc 67(337):123–129
Herrera-Semenets V, Peréz-García O, Hernández-León R (2017) Classification rule-based models for malicious activity detection. Intell Data Anal 21(5):1141–1154
Hill DR (1968) A vector clustering technique. In: Samuelson (ed) Mechanized information storage, retrieval and dissemination, pp 501–508
Ho TB (1995) An approach to concept formation based on formal concept analysis. IEICE Trans Inf Syst 78(5):553–559
Ho TB (1997) Incremental conceptual clustering in the framework of galois lattice. In: Proceedings of the first Asia-pacific conference on knowledge discovery and data mining, pp 49–64
Hotho A, Stumme G (2002) Conceptual clustering of text clusters. Proc FGML Workshop 37:37–45
Jain AK, Murty MN, Flynn PJ (1999) Data clustering: a review. ACM Comput Surv 31(3):264–323
Jamshidi Y, Kaburlasos V (2014) gsainknn: a GSA optimized, lattice computing KNN classifier. Eng Appl Artif Intell 35:277–285
Kaburlasos V, Papakostas G (2015) Learning distributions of image features by interactive fuzzy lattice reasoning (FLR) in pattern recognition applications. IEEE Comput Intell Mag 10(3):42–51
Kaburlasos V, Petridis V (2000) Fuzzy lattice neurocomputing (fln) models. Neural Netw 13(10):1145–1170
Kaburlasos V, Moussiades L, Vakali A (2009) Fuzzy lattice reasoning (flr) type neural computation for weighted graph partitioning. Neurocomputing 72(10–12):2121–2133
Kaburlasos V, Papadakis S, Papakostas G (2013) Lattice computing extension of the FAM neural classifier for human facial expression recognition. IEEE Trans Neural Netw Learn Syst 24(10):1526–1538
Katz SM (1996) Distribution of content words and phrases in text and language modelling. Nat Lang Eng 2(1):15–59
Kilander F, Jansson CG (1993) COBBIT–a control procedure for COBWEB in the presence of concept drift. In: Proceedings of the European conference on machine learning, pp 244–261
Kiran GVR, Shankar R, Pudi V (2010) Frequent itemset based hierarchical document clustering using wikipedia as external knowledge. In: Proceedings of the 14th international conference on knowledge-based and intelligent information and engineering systems: Part II, pp 11–20
Krishna SM, Bhavani SD (2010) Performance evaluation of an efficient frequent item sets-based text clustering approach. Glob J Comput Sci Technol 10(11):60–68
Kryszkiewicz M, Skonieczny L (2006) Hierarchical document clustering using frequent closed sets. Adv Soft Comput 35:489–498
Langley P, Thompson K, Iba W, Gennari JH, Allen JA (1990) An integrated cognitive architecture for autonomous agents. Tech. Rep. ARI Research Note 90-48, University of California
Lazzeroni L, Owen A (2000) Plaid models for gene expression data. Tech. rep., Stanford University
Lebowitz M (1986) Concept learning in a rich input domain: generalization-based memory. Mach Learn: An Artif Intell Approach 2:193–214
Lee J, Rajauria P, Subodh KS (2007) A model-based conceptual clustering of moving objects in video surveillance. In: Proceedings of SPIE 6506, multimedia content access: algorithms and systems
Li Y, Chung SM, Holt JD (2008) Text document clustering based on frequent word meaning sequences. Data Knowl Eng 64(1):381–404
Liu J, Wang W (2003) OP-cluster: clustering by tendency in high dimensional space. In: Proceedings of the third IEEE international conference on data mining, pp 187–194

Liu X, he P, (2005) A study on text clustering algorithms based on frequent term sets. In: Proceedings of advanced data mining and applications, LNCS vol 3584, pp 347–354
Madeira SC, Oliveira AL (2004) Biclustering algorithms for biological data analysis: a survey. IEEE/ACM Trans Comput Biol Bioinform 1(1):24–45
Malik HH, Kender JR (2006) High quality, efficient hierarchical document clustering using closed interesting itemsets. In: Proceedings of the sixth international conference on data mining, pp 991–996
Malik HH, Kender JR, Fradkin D, Moerchen F (2010) Hierarchical document clustering using local patterns. Data Min Knowl Disc 21(1):153–185
Martínez-Trinidad JF, Sánchez-Díaz G (2001) LC: a conceptual clustering algorithm. In: Proceedings of the second international workshop on machine learning and data mining in pattern recognition, pp 117–127
McKusick KB, Langley P (1991) Constraints on tree structure in concept formation. In: Proceedings of the 12th international joint conference on artificial intelligence, pp 810–816
McKusick KB, Thompson K (1990) COBWEB/3: a portable implementation. Tech. Rep. FIA-90-6-18-2, NASA Ames Research Center
Messai N, Devignes M, Napoli A, Smail-Tabbone M (2008) Many-valued concept lattices for conceptual clustering and information retrieval. In: Proceedings of the 18th European conference on artificial intelligence, pp 127–131
Michalski RS (1980) Knowledge acquisition through conceptual clustering: a theoretical framework and an algorithm for partitioning data into conjunctive concepts. A special issue on knowledge acquisition and induction. Int J Policy Anal Inf Syst 4(3):219–244
Michalski RS, Stepp RE (1983) Automated construction of classifications: conceptual clustering versus numerical taxonomy. IEEE Trans Pattern Anal Mach Intell 5(4):396–410
Michel V, Gramfort A, Varoquaux G, Eger E, Keribin C, Thirion B (2012) A supervised clustering approach for FMRI-based inference of brain states. Pattern Recogn 45(6):2041–2049. https://doi.org/10.1016/j.patcog.2011.04.006
Miller GA (1995) Wordnet: a lexical database for english. Commun ACM 38(11):39–41
Mulani N, Pawar A, Mulay P, Dani A (2015) Variant of cobweb clustering for privacy preservation in cloud db querying. Proced Comput Sci 50:363–368
Munir MU, Javed MY, Khan SA (2012) A hierarchical K-means clustering based fingerprint quality classification. Neurocomputing 85:62–67. https://doi.org/10.1016/j.neucom.2012.01.002
Ng MK, Wong JC (2002) Clustering categorical data sets using tabu search techniques. Pattern Recogn 35(12):2783–2790
Nordhausen B (1986) Conceptual clustering using relational information. In: Proceedings of the AAAI-86, pp 505–512
Papadakis S, Kaburlasos V, Papakostas G (2014) Two fuzzy lattice reasoning (FLR) classifiers and their application for human facial expression recognition. J Mult-Val Logic Soft Comput 22(4–6):561–579
Papakostas G, Savio A, Grana M, Kaburlasos V (2015) A lattice computing approach to Alzheimer's disease computer assisted diagnosis based on MRI data. Neurocomputing 150:37–42
Pei J, Zhang X, Cho M, Wang H, Yu PS (2003) MaPle: a fast algorithm for maximal pattern-based clustering. In: Proceedings of the third IEEE international conference on data mining, ICDM 2003, pp 259–266
Perner P, Attig A (2010) Fuzzy conceptual clustering. In: Proceedings of the ICDM 2010, LNAI 6171, pp 71–85
Pfitzner D, Leibbrandt R, Powers D (2009) Characterization and evaluation of similarity measures for pairs of clusterings. Knowl Inf Syst 19(3):361–394
Pons-Porrata A, Berlanga-Llavorí R, Ruiz-Shulcloper J (2002a) On-line event and topic detection by using the compact sets clustering. J Intell Fuzzy Syst 12(3–4):185–194
Pons-Porrata A, Martínez-Trinidad JF, Ruíz-Shulcloper J (2002b) RGC: a new conceptual clustering algorithm for mixed incomplete data sets. Math Comput Modell 36(11–13):1375–1385
Pons-Porrata A, Berlanga-Llavorí R, Ruiz-Shulcloper J (2007) Topic discovery based on text mining techniques. Inf Process Manage 43(3):752–768
Quan TT, Hui SC, Cao TH (2004) A fuzzy fca-based approach to conceptual clustering for automatic generation of concept hierarchy on uncertainty data. Proc CLA 2004:1–12
Ralambondrainy H (1995) A conceptual version of the K-means algorithm. Pattern Recogn Lett 16(11):1147–1157
Reich Y, Fenves SJ (1991) The formation and use of abstract concepts in design. In: Concept formation: knowledge and experience in unsupervised learning, pp 323–353
Robardet C, Feschet F (2001) Comparison of three objective functions for conceptual clustering. In: Proceedings of the 5th European conference on principles of data mining and knowledge discovery, pp 399–410

Romero-Zaliz RC, Rubio-Escudero C, Perren-Cobb J, Herrera F, Cordón O, Zwir I (2008) A multiobjective evolutionary conceptual clustering methodology for gene annotation within structural databases: a case of study on the gene ontology database. IEEE Trans Evol Comput 12(6):679–701
Sahoo N, Callan J, Krishnan R, Duncan G, Padman R (2006) Incremental hierarchical clustering of text documents. In: Proceedings of the 15th ACM international conference on information and knowledge management, pp 357–366
Scanlan J, Hartnett J, Williams R (2008) DynamicWEB: adapting to concept drift and object drift in COBWEB. In: Proceedings of the Australian conference on artificial intelligence 2008, LNAI 5360, pp 454–460
Seeman WD, Michalski RS (2006) The CLUSTER/3 system for goal-oriented conceptual clustering: method and preliminary results. In: Proceedings of the data mining and information engineering 2006 conference, pp 81–90
Segal E, Battle A, Koller D (2003) Decomposing gene expression into cellular processes. In: Proceedings of the pacific symposium on biocomputing, pp 89–100
Sen S, Adams J (2015) Real-time optimal selection of multirobot coalition formation algorithms using conceptual clustering. In: Workshops at the twenty-Ninth AAAI conference on artificial intelligence
Shi Z, Ester M (2003) Performance improvement for frequent term-based text clustering algorithm. Tech. rep., Simon Fraser University
Spanakis G, Siolas G, Stafylopatis A (2012) Exploiting wikipedia knowledge for conceptual hierarchical clustering of documents. Comput J 55(3):299–312
Stepp RE, Michalski RS (1986) Conceptual clustering: inventing goal oriented classifications of structured objects. Mach Learn: Artif Intell Approach II:471–498
Talavera L, Béjar J (2001) Generality-based conceptual clustering with probabilistic concepts. IEEE Trans Pattern Anal Mach Intell 23(2):196–206
Talmon JL, Fonteijn H, Braspenning PJ (1993) An analysis of the WITT algorithm. Mach Learn 11(1):91–104
Tanay A, Sharan R, Shamir R (2002) Discovering statistically significant biclusters in gene expression data. Bioinformatics 18:136–144
Tang C, Zhang L, Zhang I, Ramanathan M (2001) Interrelated two-way clustering: an unsupervised approach for gene expression data analysis. In: Proceedings of the second international symposium of bioinformatics and bioengineering, pp 41–48
Thompson K, Langley P (1991) Concept formation: knowledge and experience in unsupervised learning. Morgan Kaufmann, chap Concept Formation in Structured Domains, pp 127–161
Tibshirani R, Hastie T, Eisen M, Ross D, Botstein D, Brown P (1999) Clustering methods for the analysis of dna microarray data. Tech. rep., Dept. of Health Research and Policy, Dept. of Genetics, and Dept. of Biochemestry, Stanford Univ
Uddin J, Ghazali R, Deris MM, Naseem R, Shah H (2017) A survey on bug prioritization. Artif Intell Rev 47(2):145–180. https://doi.org/10.1007/s10462-016-9478-6
Velcin J, Ganascia JG (2007a) Default clustering with conceptual structures. J Data Semant VIII 4380:1–25
Velcin J, Ganascia JG (2007b) Topic extraction with AGAPE. In: Proceedings of the 3rd international conference on advanced data mining and applications, pp 377–388
Ventos V, Soldano H, Lamadon T (2004) Alpha galois lattices. In: Proceedings of the fourth ieee international conference on data mining, (ICDM04), pp 555–558
Wang H, Liu X (2011) Study on frequent term set-based hierarchical clustering algorithm. In: Proceedings of the eighth international conference on fuzzy systems and knowledge discovery (FSKD 2011), pp 1182–1186
Wang H, Pei J (2008) Clustering by pattern similarity. J Comput Sci Technol 23(4):481–496
Wang J, Han J, Pei J (2003) CLOSET+: searching for the best strategies for mining frequent closed itemsets. In: Proceedings of the international conference on knowledge discovery and data mining (KDD 2003), pp 236–245
Wang L, Tian L, Jia Y, Han W (2007) A hybrid algorithm for web document clustering based on frequent term sets and K-means. In: Advances in web and network technologies, and information management, pp 198–203
Wang TS, Lin HT, Wang P (2016) Weighted-spectral clustering algorithm for detecting community structures in complex networks. Artif Intell Rev. https://doi.org/10.1007/s10462-016-9488-4
Xu R (2014) Network intrusion detection data processing research based on concept clustering aoi algorithm. In: Applied mechanics and materials, Trans Tech Publ vol 644, pp 1162–1165
Yang J, Wang W, Wang H, Yu P (2003) Enhanced biclustering on expression data. In: Proceedings of the third IEEE conference on bioinformatics and bioengineering, pp 321–327
Yongheng W, Yan J, Shuqiang Y (2005) Parallel mining of top-K frequent items in very large text database. In: Proceedings of the 6th international conference on advances in web-age information management, pp 706–712

Yoo YP, Pettey CC, Yoo S (1996) A hybrid conceptual clustering system. In: Proceedings of the 1996 ACM 24th annual conference on computer science, pp 105–114

Yu H, Searsmith D, Li X, Han J (2004) Scalable construction of topic directory with nonparametric closed termset mining. In: Proceedings of the fourth ieee international conference on data mining, pp 563–566

Yuan G, Sun P, Zhao J, Li D, Wang C (2017) A review of moving object trajectory clustering algorithms. Artif Intell Rev 47(1):123–144. https://doi.org/10.1007/s10462-016-9477-7

Zhang W, Yoshida T, Tang X, Wanga Q (2010) Text clustering using frequent itemsets. Knowl-Based Syst 23(5):379–388

Zhao Y, Karypis G (2002) Evaluation of hierarchical clustering algorithms for document datasets. In: Proceedings of the international conference on information and knowledge management, pp 515–524

Zheng HT, Chen H, Gong SQ (2014) A frequent term-based multiple clustering approach for text documents. In: Proceedings of the 16th Asia-pacific web conference, pp 602–609

Zhuang L, Dai H (2004) A maximal frequent itemset approach for web document clustering. In: Proceedings of the fourth international conference on computer and information technology, pp 970–977