International Journal of Pattern Recognition
and Artificial Intelligence
Vol. 25, No. 3 (2011) 337–372
© World Scientific Publishing Company
DOI: 10.1142/S0218001411008683

World Scientific
www.worldscientific.com

A SURVEY OF CLUSTERING ENSEMBLE ALGORITHMS

SANDRO VEGA-PONS* and JOSÉ RUIZ-SHULCLOPER†

Advanced Technologies Application Center
7a ♯ 21812, Siboney, Havana 12200, Cuba
**svega@cenatav.co.cu*
†jshulcloper@cenatav.co.cu

Cluster ensemble has proved to be a good alternative when facing cluster analysis problems. It consists of generating a set of clusterings from the same dataset and combining them into a final clustering. The goal of this combination process is to improve the quality of individual data clusterings. Due to the increasing appearance of new methods, their promising results and the great number of applications, we consider that it is necessary to make a critical analysis of the existing techniques and future projections. This paper presents an overview of clustering ensemble methods that can be very useful for the community of clustering practitioners. The characteristics of several methods are discussed, which may help in the selection of the most appropriate one to solve a problem at hand. We also present a taxonomy of these techniques and illustrate some important applications.

Keywords: Cluster ensemble; cluster analysis; consensus partition.

1. Introduction

Cluster analysis is an essential technique in any field of research which involves analyzing or processing multivariate data, such as: data mining, taxonomy, document retrieval, image segmentation, pattern classification, etc. Its goal is to find the underlying structure of a dataset following several clustering criteria, specific properties in the data and different ways of data comparison.

A large variety of clustering algorithms has been proposed: k-Means, EM (Expectation Maximization), based on spectral graph theory, hierarchical clustering algorithms like Single-Link, Fuzzy c-Means, etc. (see Refs. 47 and 94). However, as it is known, there is no clustering method capable of correctly finding the underlying structure for all data sets.

When we apply a clustering algorithm to a set of objects, it imposes an organization to the data following an internal criterion, the characteristics of the used (dis)similarity function and the dataset. Hence, if we have two different clustering algorithms and we apply them to the same dataset, we can obtain very different results. But, which is the correct one? How can we evaluate the results? In clustering analysis, the evaluation of results is associated to the use of cluster validity indexes (CVI),[11] which are used to measure the quality of clustering results. Nevertheless, the use of the CVIs is not the definite solution. There is no CVI that impartially evaluates the results of any clustering algorithm. Thus, we can say that different solutions obtained by different clustering algorithms can be equally plausible, if there is no previous knowledge about the best way to evaluate the results. Roughly, we can assure that for any clustering algorithm there is a CVI that will evaluate satisfactorily its results.

The idea of combining different clustering results (*cluster ensemble* or *clustering aggregation*) emerged as an alternative approach for *improving* the quality of the results of clustering algorithms. It is based on the success of the combination of supervised classifiers. Given a set of objects, a cluster ensemble method consists of two principal steps: Generation, which is about the creation of a set of partitions[a] of these objects, and Consensus Function, where a new partition, which is the *integration* of all partitions obtained in the generation step, is computed.

In different articles about clustering ensemble, authors have tried to define a set of properties that endorses the use of clustering ensemble methods, such as by Fred and Jain[30] and Topchy *et al.*[76] However, which are the properties that should fulfill a clustering ensemble algorithm? There is no agreement about this unanswered question. On top of it, the verification of any of these properties in practice is very difficult due to the unsupervised nature of the clustering ensemble process. Some of them are:

- Robustness: The combination process must have better average performance than the single clustering algorithms.
- Consistency: The result of the combination should be somehow, very similar to all combined single clustering algorithm results.
- Novelty: Cluster ensembles must allow finding solutions unattainable by single clustering algorithms.
- Stability: Results with lower sensitivity to noise and outliers.

Properties like these are expected to be present in the results of a clustering ensemble process. However, the *natural* organization of data or the *ground-truth* cannot be expected as the *best* result. Moreover, it cannot be said that the clustering results obtained by a cluster ensemble method is *better* than those which were combined. It can only be ensured that the new clustering is a consensus of all the previous ones, and we can use it instead of any other clustering assuming as a fact that: the process of fusion could compensate for possible errors in a single clustering algorithm, and the decision of a group must be more reliable than any individual one.

[a] In the literature, most papers tackle the clustering problems as partitions of the representation spaces. This is not the only problem that we need to face in practice. However, in this paper, we interchangeably use the terms: partition and clustering.

This assumption is endorsed by an increasing number of applications of the clustering ensemble methods in different areas.

Over the past years, many clustering ensemble techniques have been proposed, resulting in new ways to face the problem together with new fields of application for these techniques. Despite the large number of clustering ensemble methods, there are only a few papers with the purpose of giving a summary of some of the existing clustering ensemble techniques, e.g. by Ghaemi *et al.*[31] and Li *et al.*[56] However, we think that a more general and complete study of the clustering ensemble methods is still necessary. Besides the presentation of the main methods, the introduction of a taxonomy of the different tendencies and critical comparisons among the methods is really important in order to give a practical application to a survey. Thus, due to the importance that clustering ensembles have gained facing cluster analysis problems and the amount of articles published on this topic, we have made a critical study of the different approaches and the existing methods. This paper can be very useful for the community of clustering practitioners since showing the advantages and disadvantages of each method, their implicit assumptions, can help in the selection of the appropriate clustering ensemble algorithm to solve a problem on hand.

In this paper, we will use the following uniform notation. We denote $X = \{x_1, x_2, \ldots, x_n\}$ the set of objects, where each x_i is a tuple of some α-dimensional feature space Ω^α for all $i = 1 \cdots n$. $\mathbb{P} = \{P_1, P_2, \ldots, P_m\}$ is a set of partitions, where each $P_i = \{C_1^i, C_2^i, \ldots, C_{d_i}^i\}$ is a partition of the set of objects X with d_i clusters. C_j^i is the jth cluster of the ith partition, for all $i = 1, \ldots, m$. We also denote as $\mathbb{P}_X$ the set of all possible partitions with the set of objects X, ($\mathbb{P} \subset \mathbb{P}_X$). The goal of clustering ensemble methods is to find a *consensus partition* $P^* \in \mathbb{P}_X$, which better represents the properties of each partition in $\mathbb{P}$.

The rest of this paper is organized as follows: in Sec. 2 the different clustering ensemble techniques are presented, making a differentiation between the generation mechanisms (Sec. 2.1) and the consensus functions (Sec. 2.2). Attention has been mainly paid to the different consensus functions as the fundamental step of any clustering ensemble method. In Sec. 2.3 some techniques for improving the combination quality are analyzed. A comparison of the different kinds of clustering ensemble methods is made in Sec. 2.4. Section 3 presents some applications of clustering ensemble methods and the conclusions of this research are in Sec. 4.

2. Clustering Ensembles

Every clustering ensemble method is made up of two steps: Generation and Consensus Function (see Fig. 1). The different ways of generation are described in Sec. 2.1 and in Sec. 2.2 the principal consensus function methods are discussed.

2.1. *Generation mechanisms*

Generation is the first step in clustering ensemble methods, in this step the set of clusterings that will be combined is generated. In a particular problem, it is very

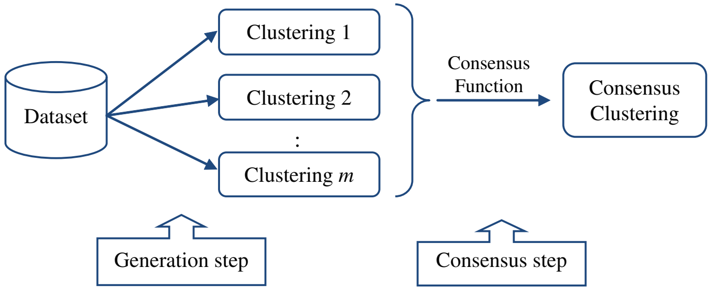

Fig. 1. Diagram of the general process of cluster ensemble.

important to apply an appropriate generation process, because the final result will be conditioned by the initial clusterings obtained in this step.

There are clustering ensemble methods like the voting-k-means[29] that demand a well-determined generation process, in this case, all the partitions should be obtained by applying the k-Means algorithm with different initializations for the number of clusters parameter. This method uses a big k value (the number of clusters), in order to obtain complex structure in the consensus partition, from the combination of small hyper-spherical structures in the single partitions.

However, in a general way, in the generation step there are no constraints about how the partitions must be obtained. Therefore, in the generation process different clustering algorithms or the same algorithm with different parameters initialization can be applied. Even different objects representations, different subsets of objects or projections of the objects on different subspaces could be used (see Fig. 2).

In the generation step the *weak clustering algorithms*[77] are also used. These algorithms make up a set of clusterings using very simple and fast procedures. Despite the simplicity of this kind of algorithms, Topchy *et al.*[78] showed that weak clustering algorithms are capable of producing high quality consensus clusterings in conjunction with a proper consensus function.

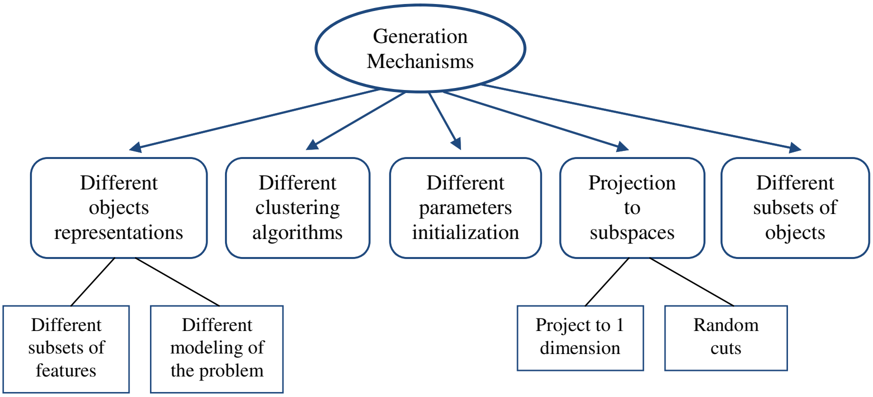
Generation Mechanisms
Different objects representations
Different clustering algorithms
Different parameters initialization
Projection to subspaces
Different subsets of objects
Different subsets of features
Different modeling of the problem
Project to 1 dimension
Random cuts

Fig. 2. Diagram of the principal clustering ensemble generation mechanisms.

In a general way, in the generation step, it is advisable to use those clustering algorithms that can yield more information about the data. It can often be very difficult to know *a priori* which clustering algorithm will be appropriate for a specific problem. The expert's experience of the problem area could be very useful in these cases. Besides, if there is no information about the problem, making a diverse cluster ensemble is recommended, since the more varied the set of partitions is, the more information for the consensus function is available. This diversity can be obtained by using the different generation mechanism presented in Fig. 2.

2.2. *Consensus functions*

The consensus function is the main step in any clustering ensemble algorithm. Precisely, the great challenge in clustering ensemble is the definition of an appropriate consensus function, capable of *improving* the results of single clustering algorithms. In this step, the final data partition or consensus partition P^*, which is the result of any clustering ensemble algorithm, is obtained. However, the consensus among a set of clusterings is not obtained in the same way in all cases. There are two main consensus function approaches: *objects co-occurrence* and *median partition*.

In the first approach, the idea is to determine which must be the cluster label associated to each object in the consensus partition. To do that, it is analyzed how many times an object belongs to one cluster or how many times two objects belong together to the same cluster. The consensus is obtained through a voting process among the objects. Somehow, each object should vote for the cluster to which it will belong in the consensus partition. This is the case, for example, of Relabeling and Voting (Sec. 2.2.1) and Co-association Matrix (Sec. 2.2.2) based methods.

In the second consensus function approach, the consensus partition is obtained by the solution of an optimization problem, the problem of finding the *median partition* with respect to the cluster ensemble. Formally, the median partition is defined as:

$$P^* = \arg\max_{P \in \mathbb{P}_X} \sum_{j=1}^{m} \Gamma(P, P_j) \tag{1}$$

where Γ is a similarity measure between partitions. The median partition is defined as the partition that maximizes the similarity with all partitions in the cluster ensemble.[b] For example, Non-Negative Matrix Factorization (Sec. 2.2.9) and Kernel (Sec. 2.2.10) based methods follow this approach.

The first mathematical treatment of the median partition problem (1) was presented by Régnier[69] (see Ref. 25 for historical details). From this moment on, several studies about the median partition problem have been made. However, the main theoretical results have been obtained for the particular case when Γ is the *symmetric difference distance* (or the *Mirkin distance*).[63] Krivanek and Moravek[52] and also Wakabayashi[88] proved by different ways that the median partition problem (1) with

[b] The median partition can be equivalently defined by minimizing the dissimilarity with respect to the cluster ensemble in the case that Γ is a dissimilarity measure between partitions.

the Mirkin distance is NP-hard. This proof was given for the case where there is a variable number of partitions m in the cluster ensemble. However, it is not known whether it is a NP-hard problem for any particular m value.[25] For $m = 1$ or $m = 2$ the solution of the problem is trivial, but for $m > 2$ nothing is known about the computational complexity.

The median partition problem with other (dis)similarity measures has not been properly studied. The complexity of the general problem is dependent on the (dis)similarity measure used in its definition. Despite the fact that the median partition problem has been proved to be NP-hard when it is defined with the Mirkin distance, we can find a (dis)similarity measure for which the problem can be solved in polynomial time. For example, defining the median partition problem with the following similarity measure

$$\Gamma'(P_a, P_b) = \begin{cases} 1, & P_a = P_b; \\ 0, & \text{otherwise.} \end{cases}$$

In the above example, the median partition can be obtained in polynomial time because one of the partitions in $\mathbb{P}$ is the solution. Indeed, if all partitions in $\mathbb{P}$ are different, the solution can be found in $O(1)$, since any partition in $\mathbb{P}$ is the solution to the problem. However, the similarity function Γ' does not have practical relevance, because it is a very *weak* similarity measure between partitions. Hence, the following question comes up. Is there any *strong*[c] similarity measure between partitions, so it allows solving the median partition problem in polynomial time? To the extent of the authors knowledge, this is an unanswered question. We think that this question has not been deeply studied and a positive answer may lead to a promising clustering ensemble technique.

Besides the Mirkin distance, there are a lot of (dis)similarity measures between partitions that can be used in the definition of the median partition problem. Deep analyses of the different (dis)similarity measures between partitions can be found in Refs. 3, 62 and 66. However, these analyses were motivated by an interest in finding the best *external cluster validity index*.[41] Therefore, the properties of these measures are not studied from the perspective of how they can be suitable for the median partition problem.

Among the main (dis)similarity measures between partitions we can find:

- Counting pairs measures. These measures count the pairs of objects on which two partitions agree or disagree. Some of them are the Rand index,[68] Fowlkes-Mallows index,[28] the Jaccard coefficient,[9] the Mirkin distance[63] and some adjusted versions of these measures.
- Set matching measures. These measures are based on set cardinality comparisons. Some of them are the Purity and Inverse Purity,[102] the F measure[82] and Dongen measure.[81]

[c] The term *strong* denotes a similarity measure that takes enough information from both partitions to determine whether they are similar or not.

- Information Theory based measures. These measures quantify the information shared between two partitions. Some of them are the Class Entropy,[8] Normalized Mutual Information,[75] Utility Function,[64] Variation of Information[62] and V-measure.[70]
- Kernel measures. These measures are defined specifically for the median partition problem and are proven to be positive semidefinite kernels.[71] Some of them are the *Graph Kernel* based measure[85] and the *Subset Significance* based measure.[86]

Consensus functions based on the *median partition approach* (1) have been theoretically more studied than the ones based on the *objects co-occurrence approach*. The median partition approach allows facing the consensus problem in a more rigorous way. In spite of that, Topchy *et al.*[79] give theoretical arguments about the validity of both approaches. They showed that the consensus solution converges to the *underlying clustering solution* as the number of partitions in the ensemble increases. However, in both approaches there are problems without definite solution, e.g. in the *object co-occurrence* approach generally, the application of a clustering algorithm as a final step to find the consensus is necessary, but the questions are: Which clustering algorithm should be used? Which are the correct parameters?

In the median partition approach, a (dis)similarity measure between clusterings is necessary, but which is the correct (dis)similarity measure? Besides, the consensus partition is usually defined as the optimum of an exponential optimization problem; however, which is the best heuristic to solve the problem or to come close to the solution?

A lot of clustering ensemble methods have been proposed in recent years trying to answer questions like the previous ones. The consensus problem has been faced by using several mathematical and computational tools. Methods based on Relabeling and Voting, Co-association Matrix, Graph and Hypergraph partitioning, Mirkin distance, Information Theory, Finite Mixture Models, Genetic Algorithms, Locally Adaptive Clustering Algorithms (LAC), Kernel methods, Non-Negative Matrix Factorization (NMF) and Fuzzy techniques can be found. In Fig. 3, a taxonomy of the main consensus functions is presented.

In Fig. 3, besides the taxonomy based on the mathematical or computation tool used in each kind of clustering ensemble technique, a correspondence between each kind of technique and one of the two consensus function approaches defined above (object co-occurrence or median partition) is presented. In principle, this correspondence between these two taxonomies of consensus functions does not have to be unique, e.g. there could be two consensus clustering methods based on genetic algorithms, one following the co-occurrence approach and the other, the median partition approach. However, we made explicit this correspondence since it actually holds in practice. On the other hand, some consensus functions present the peculiarity that they are defined through the median partition problem, but in practice, the consensus partition is obtained by means of a mechanism related with the object co-occurrence approach. These are the cases, for instance, of the Graph

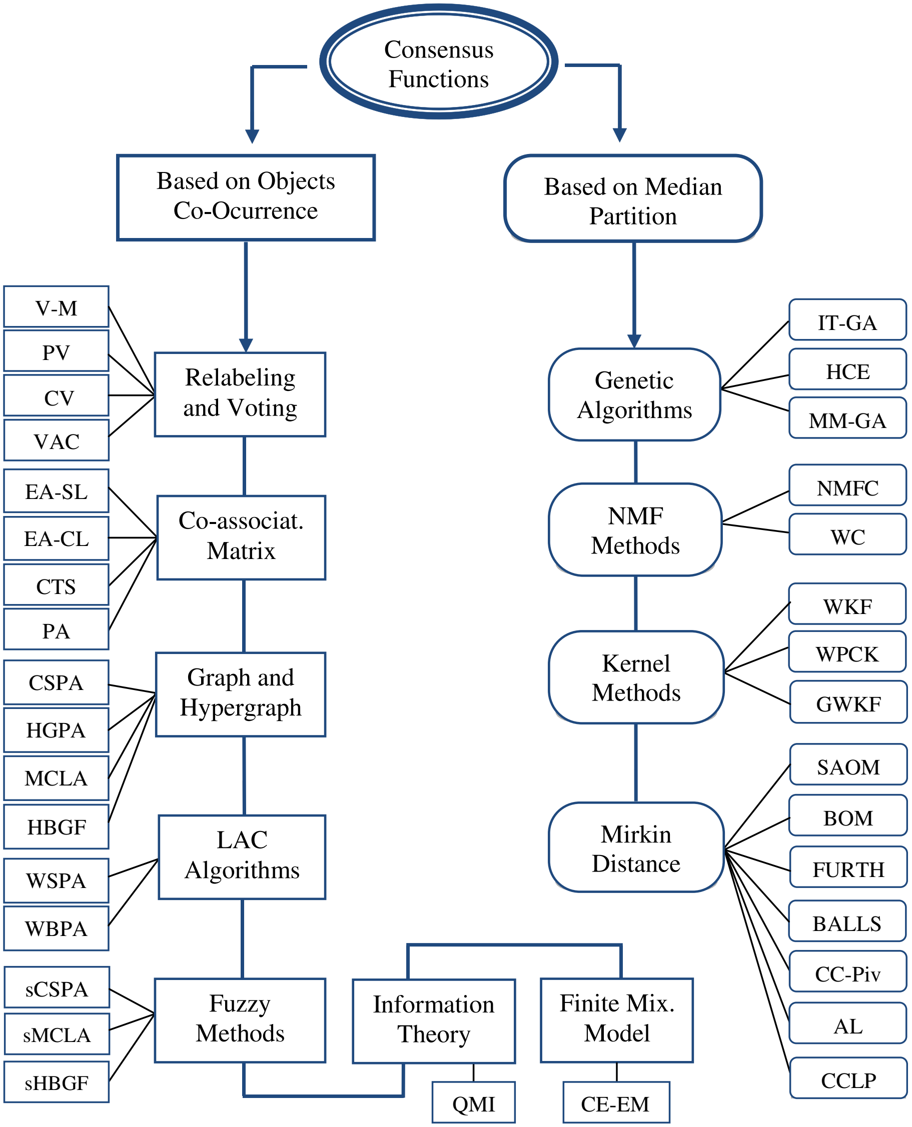

Fig. 3. Diagram of the principal consensus functions techniques. Consensus functions based on object co-occurrence approach are represented by a rectangle (left) and the ones based on the median partition approach are represented by a rounded rectangle (right).

and Hypergraph based methods (Sec. 2.2.3) and Information Theory based methods (Sec. 2.2.5). We put these algorithms in the object co-occurrence classification in Fig. 3.

In the next sections, we will present an analysis of each kind of clustering ensemble methods. In this analysis, we will explain the most popular clustering ensemble techniques. Also, for each kind of method, we will talk about its strength and weakness for defining the clustering ensemble problem, as well as their advantages and drawbacks for obtaining the consensus partition.

2.2.1. *Relabeling and Voting based methods*

The Relabeling and Voting methods are based on solving as first step the *labeling correspondence problem* and after that, in a voting process, the consensus partition is obtained. The labeling correspondence problem consists of the following: the label associated to each object in a partition is symbolic; there is no relation between the set of labels given by a clustering algorithm and the set of labels given by another one. The label correspondence is one of the main issues that makes unsupervised combination difficult. The different clustering ensemble methods based on relabeling try to solve this problem using different heuristics such as *bipartite matching* and *cumulative voting*. A general formulation for the voting problem as a multi-response regression problem was recently presented by Ayad and Kamel.[6] Among the relabeling based methods Plurality Voting (PV),[26] Voting-Merging (V-M),[18] Voting for fuzzy clusterings,[19] voting Active Clusters (VAC),[80] Cumulative Voting (CV)[7] and the methods proposed by Zhou and Tang[103] and Gordon and Vichi[36] are found.

Dudoit and Fridlyand[22] and Fischer and Buhmann[26] presented a voting consensus algorithm similar to plurality voting used in supervised classifiers ensembles.[55] In this method, it is assumed that the number of clusters in each partition is the same and equal to the final number of clusters in the consensus partition. The labeling correspondence problem is solved through a maximum-likelihood problem using the Hungarian[53] method. After that, a plurality voting procedure is applied to obtain the winner cluster for each object.

In the Voting-Merging[18] method, a way of combining clustering using a voting schedule is proposed. This consists of two main steps: a voting process and the merging of the votes. In the voting step, it is necessary to solve the label correspondence problem. After that, for each object x_i and each cluster C_j, $D_j(x_i)$ is defined as the number of times that the jth label is associated to x_i. At this moment, a fuzzy partition of the set of objects X is obtained. Then, to each object x_i, the cluster $C_g = \arg\max_{C_j}\{D_j(x_i)\}$ is assigned. On the other hand, $\pi(C_q, C_j) = \mathrm{mean}_{x\in C_q}\{D_j(x)\}$ is defined as a measure of how much the elements in the qth cluster belong to the jth cluster. Therefore, $\pi(C_q,\ C_j)$ is a non-symmetric measure of the C_q, C_j neighborhood relation. It is said that the cluster C is the closest one to C_q if $\pi(C_q, C) = \max_{j\neq q}\{\pi(C_q, C_j)^2\}$. All pairs of clusters C_q, C_j such that C_q is the closest cluster to C_j and C_j is the closest cluster to C_q are merged using this measure. A chain of clusters $(C_{q_1}, C_{q_2}, \ldots, C_{q_h})$, in which each cluster C_{q_i} is the closest to its consecutive $C_{q_{i+1}}$ and the last cluster C_{q_h} is the closest to the first one in the chain C_{q_1}, is also merged.

The Voting Active Clusters[80] method provides an adaptive voting method where the votes are updated in order to maximize an overall quality measure. This method allows the combination of clusterings from different locations, i.e. all the data does not have to be collected in one central work station. The idea is to make different clusterings from different portions of the original data in separate processing centers. Afterwards, the consensus clustering is obtained through a voting mechanism.

If there exists a relation among the labels associated for each clustering algorithm, the voting definition of the clustering ensemble problem would be the most appropriate. However, the labeling correspondence problem is what makes the combination of clusterings difficult. This correspondence problem can only be solved, with certain accuracy, if all partitions have the same number of clusters. We consider this to be a strong restriction to the cluster ensemble problem. Then, in general, they are not recommended when the number of clusters in all partitions in the ensemble is not the same. Besides, very frequently, they could have high computational cost since the Hungarian algorithm to solve the label correspondence problem is $\mathcal{O}(k^3)$, where k is the number of clusters in the consensus partition. On the other hand, these kinds of algorithms are usually easy to understand and implement.

2.2.2. *Co-association matrix based methods*

The idea of co-association is used to avoid the label correspondence problem. Co-association methods (see Ref. 30), map the partitions in the cluster ensemble into an intermediate representation: the co-association matrix. Each cell in the matrix has the following value:

$$CA_{ij} = \frac{1}{m} \sum_{t=1}^{m} \delta(P_t(x_i), P_t(x_j)) \tag{2}$$

where $P_t(x_i)$ represents the associated label of the object x_i in the partition P_t, and $\delta(a, b)$ is 1, if $a = b$, and 0 otherwise. Then, the value in each position (i, j) of this matrix is a measure about how many times the objects x_i and x_j are in the same cluster for all partitions in $\mathbb{P}$. This matrix can be viewed as a new similarity measure between the set of objects X. The more objects x_i and x_j appear in the same clusters, the more similar they are. Using the co-association matrix CA as the similarity measure between objects, the consensus partition is obtained by applying a clustering algorithm.

In Ref. 29, a fixed threshold equal to 0.5 is used to generate the final consensus partition. It is obtained by joining in the same cluster, objects with a co-association value greater than 0.5.

Fred and Jain[30] proposed a modification where an algorithm is applied to find a minimum spanning tree, after obtaining the co-association matrix, i.e. seeing the co-association matrix as an adjacency matrix of a graph, a tree that contains all the nodes of the graph and the minimum weights in their edges are searched. Then, the weak links between nodes are cut using a threshold r. This is equivalent to cutting the dendrogram produced by the Single Link (SL)[47] algorithm using the threshold r. This threshold is obtained by using a simple but effective heuristic called *highest lifetime* criterion. In Ref. 30 the *k-cluster lifetime* is defined as the range of threshold values on the dendrogram to obtain k clusters. After computing the lifetime value of each level, the one with the highest value is selected as the final partition of the data. Besides, the Complete-Link (CL), Average-Link (AL) and other hierarchical clustering algorithms can be used as variants of this method.

Li *et al.*[57] introduced a new hierarchical clustering algorithm that is applied to the co-association matrix to improve the quality of the consensus partition. This algorithm is based on the development of the concept of normalized edges to measure similarity between clusters.

In the co-association matrix (2), $\delta(a, b)$ takes only the values 0 or 1. That way, the new similarity between objects is computed only by taking into account whether the two objects belong to the same cluster or not. We think that a representation, which uses additional information to make the similarity measure should be more expressive about the real relationship between the objects.

In this direction, two similarity matrixes: Connected-Triple Based Similarity (CTS) and SimRank Based Similarity (SRS) are proposed by Iam-on *et al.*[46] The CTS works on the basis that if two objects share a link with a third object, then this is indicative of similarity between those two objects. The SRS reflects the underlying assumption that neighbors are similar if their neighbors are similar as well. Also, Vega-Pons and Ruiz-Shulcloper[87] presented the Weighted Co-Association Matrix, which computes the similarity between objects using the size of the cluster, the number of clusters in each partition and the original similarity values between the objects. Besides, Wang *et al.*[90] introduced the *Probability accumulation matrix*, which is conformed taking into account the size of clusters, as well as the number of features in the object representation. These matrixes take into account more information than the traditional co-association (2) and they can measure the pair-wise correlation between objects in higher resolution.

All the co-association methods are based on the construction of a new similarity measure between objects from the clustering ensemble. Also, a clustering algorithm to obtain the final partition is necessary. Hence, the consensus clustering will be conditioned by the way that the similarity is created and the particular algorithm applied (and its parameters initialization). Besides, this kind of algorithms have a computational complexity of $\mathcal{O}(n^2)$, and cannot be applied to large datasets. However, they are very easy to implement and understand.

2.2.3. *Graph and hypergraph based methods*

This kind of clustering ensemble methods transform the combination problem into a graph or hypergraph partitioning problem. The difference among these methods lies on the way the (hyper)graph is built from the set of clusterings and how the cuts on the graph are defined in order to obtain the consensus partition.

Strehl and Ghosh[75] defined the consensus partition as the partition that most information shares with all partitions in the cluster ensemble. To measure the information shared by two partitions, the Normalized Mutual Information (NMI) is used based on the *Mutual Information* concept from Information Theory.[15] The NMI is a similarity measure between partitions defined as follows:

Let $P_a = \{C_1^a, C_2^a, \ldots, C_{d_a}^a\}$ and $P_b = \{C_1^b, C_2^b, \ldots, C_{d_b}^b\}$ be two partitions of X, d_a being the number of clusters in P_a and d_b the number of clusters in P_b. Let n_{ia} be the number of objects in the ith cluster of the partition P_a, n_{bj} the number of objects in the jth cluster of the partition P_b and n_{ij} the number of objects which are together in the ith cluster of the partition P_a and in the jth cluster of the partition P_b. The *Normalized Mutual Information* between P_a and P_b is expressed in the following way:

$$\mathrm{NMI}(P_a, P_b) = \frac{-2\sum_{i=1}^{d_a}\sum_{j=1}^{d_b}\frac{n_{ij}}{n}\log(\frac{n_{ij}\cdot n}{n_{ia}\cdot n_{bj}})}{\sum_{i=1}^{d_a} n_{ia}\log(\frac{n_{ia}}{n}) + \sum_{j=1}^{d_b} n_{bj}\log(\frac{n_{bj}}{n})}$$

It takes 1 as a maximum value and 0 as a minimum.

This way, the consensus partition is defined as:

$$P^* = \arg\max_{P\in\mathbb{P}_X}\sum_{j=1}^{m}\mathrm{NMI}(P, P_j) \tag{3}$$

where $\mathbb{P}_X$ is the set of all possible partitions with the set of objects X.

An exhaustive search to solve this problem is computationally intractable. To face this problem three heuristics based on hypergraph partitioning are proposed by Strehl and Ghosh,[75] CSPA, HGPA and MCLA. The three heuristics start from representing the clustering ensemble as a hypergraph, where each partition is represented by an hyperedge.

In the *Cluster-based Similarity Partitioning Algorithm* (CSPA), from the hypergraph, a $n \times n$ similarity matrix (the co-association matrix) is constructed. This can be viewed as the adjacency matrix of a fully connected graph, where the nodes are the elements of the set X and an edge between two objects has an associated weight equal to the number of times the objects are in the same cluster. After that, the graph partitioning algorithm METIS[49] is used for obtaining the consensus partition.

The *HyperGraphs Partitioning Algorithm* (HGPA) partitions the hypergraph directly, by eliminating the minimal number of hyperedges. It is considered that all hyperedges have the same weight, and it is searched by cutting the minimum possible number of hyperedges that partition the hypergraph in k connected components of approximately the same dimension. For the implementation of the method, the hypergraphs partitioning package HMETIS[48] is used.

In the *Meta-CLustering Algorithm* (MCLA), first of all the similarity between two clusters C_i and C_j is defined in terms of the amount of objects grouped in both, using the Jaccard index.[9] Then, a matrix of similarity between clusters is formed, which represents the adjacency matrix of the graph built considering the clusters as nodes and assigning a weight to the edge between two nodes, equal to the similarity between the clusters. After that, this graph is partitioned using METIS[49] algorithm and the obtained clusters are called meta-clusters. Finally, to find the final partition, the times each object appears in a meta-cluster is calculated and each object is assigned to the meta-cluster to which it is assigned more times. This algorithm is quadratic with respect to the amount of clusters in the set of partitions $\mathbb{P}$, which in most applications is significantly smaller than n^2.

Another method, the *Hybrid Bipartite Graph Formulation* (HBGF) was introduced by Fern *et al.*[23] It models the clusters and the objects together in the same graph. In this method, a bipartite graph is built where there are no edges between vertexes that are both either instances or clusters. There only exists an edge between two nodes if one of the nodes represents a cluster and the other node represents an object that belongs to this cluster. The consensus partition is obtained partitioning this graph by using the METIS[49] algorithm or the Spectral clustering.[65]

Recently, Abdala *et al.*[1] proposed a graph based clustering ensemble algorithm based on the random walker algorithm for the combination of image segmentations.[91] First, given a parameter δ, a graph that connects each object with its δ nearest neighbors is built. In this graph, a weight is associated to each edge according to the co-association (2) value between the objects that compose the edge. Then, some seed regions are automatically generated in the graph and using these seed regions the random walker algorithm[37] is applied to obtain the consensus result.

The graph and hypergraph based methods are among the most popular methods. They are easy to understand and implement. Moreover, in most cases they have low computational complexity (less than quadratic in the number of objects), for example, HGPA ($\mathcal{O}(k \cdot n \cdot m)$), MCLA ($\mathcal{O}(k^2 \cdot n \cdot m^2)$) and HBGF ($\mathcal{O}(k \cdot n \cdot m)$), where n is the number of objects, m the number of partitions and k the number of clusters in the consensus partition. Only the CSPA method has a computational and storage complexity of $\mathcal{O}(k \cdot n^2 \cdot m)$, which is quadratic in the number of objects. We put more attention in the complexity respect to the number of objects n, because in practice, $m \ll n$ and k almost always takes relatively small values.

We consider that the main weakness of this kind of clustering ensemble methods is that they are not rigourously well-founded as a solution for the consensus clustering problem, in the sense that most of them are proposed as a solution for the median partition problem (3) defined with the NMI similarity measure, but in practice, they are not solving this problem. These methods are more related with the *object co-occurrence approach* since in the (hyper)graph construction and in the partitioning algorithm, the relationship between individual objects are implicitly taken into account. In addition to that, these methods need a (hyper)graph partitioning algorithm in the final step, therefore, if we change this algorithm, the final result could change. Regardless of the fact that METIS and HMETIS are the most used algorithm for the (hyper)graph partitioning, they are not the only graph partitioning algorithm and they do not have to achieve the best results in all situations.

2.2.4. *Mirkin distance based methods*

Given two partitions P_a and P_b of the same dataset X, the following four categories are defined:

- n_{00}: The number of pairs of objects that were clustered in separate clusters in P_a and also in P_b.
- n_{01}: The number of pairs of objects that were clustered in different clusters in P_a, but in the same cluster in P_b.
- n_{10}: The number of pairs of objects that were co-clustered in the same cluster in P_a, but not in P_b.
- n_{11}: The number of pairs of objects that were co-clustered in both partitions.

The *symmetric difference distance* or Mirkin distance $\mathcal{M}$ is defined as $\mathcal{M}(Pa, P_b) = n_{01} + n_{10}$, which represents the number of disagreements between the two partitions. The median partition problem defined with this similarity measure (4) was proven to be a NP-complete problem.

$$P^* = \arg \min_{P \in \mathbb{P}_X} \sum_{j=1}^{m} \mathcal{M}(P, P_j) \tag{4}$$

Some methods were proposed to get an exact solution of this problem, e.g. Refs. 39 and 89. However, they can only be applied in small instances of this problem, i.e. with small numbers of objects and partitions.

Thus, several heuristics have been proposed to face this problem, in some cases with a known approximation factor.[84] In Ref. 25, three heuristics to solve this problem were introduced.

The first one, called *Best-of-k* (BOK), is a very simple heuristic which consists of selecting the partition $P \in \mathbb{P}$ closest to the solution of the problem (4). In other words, the output of this heuristic is the partition in the cluster ensemble that minimizes the distance from it to all the other partitions in the ensemble. For the Mirkin distance case, this simple solution is proven to be a factor 2-approximation of the median partition problem.

The second heuristic, *Simulated Annealing One-element Move* (SAOM), follows the idea of guessing an initial partition and iteratively changing it by moving one object from one cluster to another. This way, better solutions could be found. The initial partition can be randomly selected or, for example, the output of the BOK algorithm. In the SAOM heuristic the simulated annealing[51] meta-heuristic is applied in order to avoid the convergence to a local optimum. The results of this algorithm are dependent on the selection of the initial partition and the global parameter selected for the simulated annealing meta-heuristic.

The third heuristic, *Best One-element Move* (BOM), is based on the previous idea of starting with an initial partition and after that generating new partitions by moving object from one cluster to another. In this case, a greedy process is followed.

At each step, if a better partition exists, it is taken as the new solution. This algorithm is much more dependent on the initial partition than the previous one (SAOM). An improper selection of the initial partition could lead to a fast convergence to a poor quality local optimum.

Gionis *et al.*[32] proposed four new heuristics to solve the problem (4). The first one is called *Balls algorithm*. In this algorithm, a graph is built with the objects, where the edges are weighted by the distances between pairs of objects. Based on the triangle inequality of the Mirkin distance, an iterative process is applied, where in each iteration a new cluster for the consensus partition is obtained. To obtain the new clusters, for each object x, a set B_x with the objects at a distance of at most $1/2$ from x is selected. If the average distance of the nodes in B_x to x is less or equal to a parameter λ, the objects $B_x \cup \{x\}$ are considered a new cluster; otherwise, the object x forms a singleton cluster.

The second algorithm, *Agglomerative algorithm*, works like the standard Average-Link agglomerative clustering algorithm. Starting with any object forming a singleton cluster, if the average distance of the closest pair of clusters is less than $1/2$ the clusters are merged. When there are not two clusters with an average distance less than $1/2$ the algorithm stops and the current solution is given as the proposed median partition.

The third algorithm, *Furthest algorithm*, is a top-down heuristic. Starting with all the objects in the same single cluster, in each step, the pair of nodes furthest apart are placed in different clusters and the cost of the new partition is computed. The cost of a partition is computed as the sum of all distances from it to all partitions in the cluster ensemble. This procedure is repeated until a worse solution is obtained.

The fourth algorithm, *LocalSearch algorithm*, is a local search heuristic. This algorithm starts with an initial partition, which is iteratively changed. In this algorithm, a particular way of computing the cost of changing an object from one cluster to another is defined. Thus, the idea of the algorithm is to change the objects in the clusters and this process is repeated until there is no move that can improve the cost. This algorithm can be applied as a clustering algorithm or as a post-processing step to improve the quality of the previous algorithms.

These four heuristics have a computational complexity quadratic in the number of objects. Then, they are not scalable to large datasets.

Bertolacci and Wirth[10] made an examination of some approximation algorithms for the consensus clustering problem. Besides the algorithms previously presented in this section, two other algorithms were used: *CC-Pivot* and *CCLP-Pivot*. These algorithms were first proposed by Alion *et al.*[2]

The *CC-Pivot* is based on the idea of the well-known sorting algorithm *Quicksort*. In the CC-Pivot algorithm, pivot objects are repeatedly selected and a partition is obtained by means of the relation between the objects and each pivot. The pivots are usually randomly selected, however other heuristics can be used for the selection.[83] This method has a computational complexity of $\mathcal{O}(k \cdot n \cdot m)$ (k is the number of clusters in the final partition, n the number of objects and m the number of partitions).

The *CCLP-Pivot* is a linear programming based version of the CC-Pivot. However, *CCLP-Pivot* has a computational complexity of $\mathcal{O}(n^3)$, which is extremely high to consider this algorithm a possible way of facing the consensus clustering problem.

Goder and Filkov[34] made an experimental comparison of these previous techniques and came to the conclusion that clustering based heuristics are generally faster while local search heuristics give generally better results.

In these methods, the consensus partition is obtained by the solution of the median partition problem when using the Mirkin distance as dissimilarity measure between partitions (4). The Mirkin distance is the most studied measure for the median partition problem. However, in practice, it does not have to be the most appropriate for all situations. Some of the heuristics previously discussed have a high computational complexity, thus they cannot be applied to large datasets. However, most of them are easy to understand and program.

2.2.5. *Information Theory based methods*

Another method to solve the optimization problem (1) was introduced by Topchy *et al.*[78] The *category utility function*[33] $U(P_h, P_i)$ is defined as a similarity measure between two partitions $P_h = \{C_1^h, C_2^h, \ldots, C_{d_h}^h\}$ and $P_i = \{C_1^i, C_2^i, \ldots, C_{d_i}^i\}$ as follows:

$$U(P_h, P_i) = \sum_{r=1}^{d_h} \rho(C_r^h) \sum_{j=1}^{d_i} \rho(C_j^i|C_r^h)^2 - \sum_{j=1}^{d_i} \rho(C_j^i)^2 \tag{5}$$

where $\rho(C_r^h) = \frac{|C_r^h|}{n}$, $\rho(C_j^i) = \frac{|C_j^i|}{n}$ and $\rho(C_j^i|C_r^h) = \frac{|C_j^i \cap C_r^h|}{|C_r^h|}$.

In this case, the category utility function can be interpreted as the difference between the prediction of the clusters of a partition P_i both with the knowledge of the partition P_h and without it. This way, the better agreement between the two partitions, the higher values of the category utility function we shall have.

Hence the consensus partition could be defined by using U as a similarity measure between partitions:

$$P^* = \arg\max_{P \in \mathbb{P}_X} \sum_{i=1}^{m} U(P, P_i)$$

It has been proved[64] that this utility function is equivalent to within-cluster variance minimization, then it can be maximized by applying the k-Means algorithm. Using a generalized definition of entropy,[15] the utility function is transformed into the normalized mutual information. Finally, this method offers the same criterion of consensus that the normalized mutual information (NMI) and the k-Means algorithm can be used as a heuristic solution.

This algorithm defines the consensus problem as the search of the median partition and a heuristic solution is proposed. In this method, the category utility function is used as the similarity measure between partitions. However, the heuristic proposed to obtain the consensus partition uses the k-Means algorithm to determine the label associated to each object in the final partition. Thus, this method is actually more related with the object co-occurrence approach than with the median partition approach. On the other hand, the final partition is conditioned by the structure imposed on the data by the k-Means algorithm, i.e. clusters with hyper-spherical shape, the number of clusters in the final partition must be specified, among others. Besides, this method requires to be restarted several times to avoid the convergence to low quality local minima. However, the computational complexity of this method is very low, $\mathcal{O}(k \cdot n \cdot m)$ (k is the number of clusters in the final partition, n the number of objects and m the number of partitions), which is generally significantly smaller than n^2.

2.2.6. *Finite mixture models based methods*

Topchy *et al.*[76] proposed a new consensus function, where the consensus partition is obtained as the solution of a maximum likelihood estimation problem. The problem of maximum likelihood is solved by using the EM[61] algorithm.

This consensus approach is based on a finite mixture model for the probability of assigning labels to the objects in the partitions. The main assumption is that the labels y_i (label assigned to the object x_i) are modeled as random variables drawn from a probability distribution described as a mixture of multivariate component densities:

$$\rho(y_i|\Theta) = \sum_{t=1}^{k} \lambda_t \rho_t(y_i|\theta_t) \tag{6}$$

where each component is parameterized by θ_t. The k components in the mixture are identified with the k clusters of the consensus partition $P^* = \{C_1, C_2, \dots, C_k\}$. The mixing coefficients λ_t correspond to the prior probabilities of the clusters. All the data $Y = \{y_i\}_{i=1}^{n}$ is assumed to be independent and identically distributed.

This allows representing the logarithmic likelihood function to the parameters $\Theta = \{\lambda_1, \dots, \lambda_k, \theta_1, \dots, \theta_k\}$ given the set of data Y as:

$$\log L(\Theta|\mathrm{Y}) = \log \prod_{i=1}^{n} \rho(y_i, \Theta) = \sum_{i=1}^{n} \log \sum_{t=1}^{k} \lambda_t \rho_t(y_i|\theta_t) \tag{7}$$

The searching of the consensus partition is formulated as a problem of maximal likelihood estimation:

$$\Theta^* = \arg \max_{\Theta} \{\log L(\Theta|Y)\}$$

The maximal likelihood problem (7) generally cannot be solved in a closed form (just in terms of functions and elemental operations) when the parameters Θ are unknown. However, the likelihood function (6) can be optimized by using the EM algorithm, assuming the existence of hidden data Z and the likelihood of complete data (Y, Z).

To do that, it is started with some arbitrary initial parameters $\{\lambda'_1, \ldots, \lambda'_k, \theta'_1, \ldots, \theta'_k\}$. After that, an iterative process given by two steps: Expectation (E) and Maximization (M), is repeated until a convergence criterion is satisfied.

The E-step calculates the expected values of the hidden variables and the M-step maximizes the likelihood calculating a new and better parameter estimation. The convergence criteria can be based on the increase in the amount of likelihood function between two consequent M-steps.

The consensus partition is obtained by a simple inspection of the expected values of the hidden variables $E[z_{it}]$ because $E[z_{it}]$ represents the probability that pattern y_i was generated by the tth mixture component, which represents the tth cluster. When some convergence criterion is achieved, label y_i is assigned to the component that has the largest value of the hidden variable.

In this method, the data is modeled as random variables and it is assumed that they are independent and identically distributed, which are three restrictions to the general problem. Besides, the number of clusters in the consensus partition must be fixed because it is necessary to know the number of components in the mixture model. However, this method has a low computational complexity $\mathbb{O}(k \cdot n \cdot m)$ comparable with the k-means algorithm.

2.2.7. *Genetic algorithms based methods*

These methods use the search capability of genetic algorithms to obtain the consensus clustering. Generally, the initial population is created with the partitions in the cluster ensemble and a fitness function is applied to determine which chromosomes (partitions of the set of object) are closer to the clustering than it is searching for. After that, crossover and mutation steps are applied to obtain new offsprings and renovate the population. During this process, if any termination criterion is achieved, the partition with the highest fitness value is selected as the consensus partition.

Among the methods based on genetic algorithms the Heterogeneous Clustering Ensemble, Refs. 97 and 98 can be found. The initial population in this method is obtained by using any kind of generation mechanisms. With each pair of partitions obtained from the objects, an ordered pair is created. The process of reproduction uses a fitness function as a unique way to determine if a pair of partitions (chromosomes) will survive or not in the next stage. In this algorithm, the fitness function is produced for the comparison of the amount of overlaps between the partitions in each chromosome.

For a pair of partitions (P, P'), first, the overlapping of P with respect to P' is computed. For each cluster C in P the cluster of P' which most objects shares with C is searched, and the number of overlapping objects is counted. After that, the overlapping value of P' with respect to P is computed in the same way. Finally, the fitness function for this pair is equal to the sum of these overlapping values. This function gives a representative value to each pair of partitions, and the crossover is applied to the pair that has the highest value.

In the crossover process, offspring are obtained from the selected pair, keeping the most possible amount of information from the parents in the newly obtained partitions. Later, the parent partitions are replaced by their offspring and another iteration of the complete algorithm is applied.

Another clustering ensemble method based on genetic algorithms is the method proposed by Luo *et al.*,[60] where the consensus clustering is found by minimizing an information theoretical criterion function using a genetic algorithm. This method (called IT-GA in Fig. 3) uses a metric between clusterings based on the entropy between partitions. It also uses the Hungarian method to solve the label correspondence problem. Another clustering ensemble method based on genetic algorithms was proposed by Analoui and Sadighian.[4] In this paper, the consensus is proposed through a probabilistic model by using a finite mixture of multinomial distributions. The final result is found as a solution to the corresponding maximum likelihood problem using a genetic algorithm.

The search capabilities of genetic algorithms is used in these methods. It allows exploring partitions that are not easy to be found by other methods. However, a drawback of these algorithms is that a solution is *better* only in comparison to another; such an algorithm actually has no concept of an *optimal solution*, or any way to test whether a solution is optimal or not. Besides, successive runs of this kind of clustering ensemble algorithms may produce very different results, due to its extremely heuristic nature.

2.2.8. *Locally adaptive clustering algorithm based methods*

This kind of consensus function combines partitions obtained by using locally adaptive clustering algorithms (LAC).[21] When a LAC algorithm is applied to a set of objects X, it gives as an output a partition $P = \{C_1, C_2, \dots, C_q\}$, which can be also identified by two sets $\{\mathsf{c}_1, \dots, \mathsf{c}_q\}$ and $\{w_1, \dots, w_q\}$, where c_i and w_i are the centroid and the weight associated to the cluster C_i respectively. The LAC algorithms are designed to work with numerical data, i.e. this method assumes that the object representation in the dataset is made up of numerical features: $X = \{x_1, \dots, x_n\}$, with $x_j \in \mathbb{R}^\alpha$, $j = 1, \dots, n$. Also, $\mathsf{c}_i \in \mathbb{R}^\alpha$ and $w_i \in \mathbb{R}^\alpha$, $i = 1, \dots, k$. The set of partitions $\mathbb{P} = \{P_1, P_2, \dots, P_m\}$ is generated by applying LAC algorithms m times with different parameters initialization.

Domeniconi and Al-Razgan[20] proposed two principal consensus functions. The first one is named *Weighty Similarity Partition Algorithm* (WSPA). For each object x_i the weighted distance to each cluster C_t for a partition $P = \{C_1, C_2, \dots, C_q\}$ in $\mathbb{P}$ is computed by:

$$d_{it} = \sqrt{\sum_{s=1}^{l} w_{ts}(x_{is} - \mathsf{c}_{ts})^2} \tag{8}$$

where x_{is} represents the sth attribute value of x_i, c_{ts} is the sth attribute value of the centroid c_t and w_{ts} is the weight value assigned to the sth attribute in the cluster C_t.

Let $D_i = \max_t\{d_{it}\}$ be the maximal distance from x_i to all the clusters. Assuming that the cluster label assigned to each object x_i is a random variable from a distribution, with probabilities $\{\rho(C_1|x_i), \rho(C_2|x_i), \ldots, \rho(C_q|x_i)\}$, it is defined $\rho(C_t|x_i) = \frac{D_i - d_{it} + 1}{q \cdot D_i + q - \sum_t d_{it}}$ where the denominator is useful as a normalization factor which guarantees that $\sum_{t=1}^{q} \rho(C_t|x_i) = 1$. That way, the posterior probability vector associated to each object x_i is built:

$$\rho_i = (\rho(C_1|x_i), \quad \rho(C_2|x_i), \ldots, \rho(C_q|x_i))^T$$

where T denotes the transpose of the vector.

To calculate the similarity between two objects x_i, x_j the similarity of the cosine between the vectors associated to these objects is used, that is:

$$cs(x_i, x_j) = \frac{\rho_i^T \rho_j}{\|\rho_i\| \, \|\rho_j\|}$$

Next, all the similarities between objects are combined in a matrix $\mathcal{S}$ of $n \times n$, where $\mathcal{S}_{ij} = cs(x_i, x_j)$. As there are m partitions in $\mathbb{P}$, m similarity matrixes $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_m$ can be made. Let Ψ be the combination of these matrixes:

$$\Psi = \frac{1}{m} \sum_{r=1}^{m} \mathcal{S}_r$$

where each entry of this matrix Ψ_{ij} reflects the similarity average between the objects x_i and x_j. As the last step, a graph $G = (V, E)$ is built where each vertex v_i identifies the object x_i and the edge e_{ij} that connects the vertexes v_i and v_j has a value equal to Ψ_{ij}. The METIS algorithm is used to obtain the consensus partition from this graph.

The second consensus function introduced by Domeniconi and Al-Razgan[20] is the *Weighty Bipartite Partition Algorithm* (WBPA). As its name indicates, it presents a consensus function based on the partitioning of a bipartite graph. Following the steps of the WSPA algorithm, for an object x_i and a partition P_v, $v = 1, \ldots, m$, the posterior probabilities vector ρ_i^v is computed. With these vectors, the matrix A is built as following:

$$A = \begin{pmatrix} (\rho_1^1)^T & \cdots & (\rho_1^m)^T \\ \vdots & \ddots & \vdots \\ (\rho_n^1)^T & \cdots & (\rho_n^m)^T \end{pmatrix}$$

where $(\rho_i^v)^T$ are row vectors. The dimensionality of A is $n \times q \cdot m$, assuming that each one of the m applied algorithms produces q clusters. Based on A, a bipartite graph is defined, which will be partitioned to obtain the consensus partition.

Let $G = (V, E)$ be the graph with $V = V^c \cup V^I$, where V^c contains $q \cdot m$ vertexes, where each one represents a cluster and V^I contains n vertexes, each one associated to an object. The edges will have a weight associated in the following way:

(1) $E(i,j)=0$. In the case in which both vertexes v_i, v_j, represent objects or both represent clusters.

(2) $E(i,j)=A(i-q\cdot m, j)$. In the case that v_i represents an object and v_j represents a cluster or vice versa.

A partition of the bipartite graph G would group the objects and clusterings simultaneously and the consensus partition can be obtained from the objects partitioning.

Also, Domeniconi and Al-Razgan[20] proposed a third heuristic called *Weighted Subspace Bipartite Partitioning Algorithm* (WSBPA), which is basically the WBPA heuristic but, it adds a weight vector to each cluster in the output of the algorithm.

This kind of methods imposes a strong restriction to the clustering ensemble problem meaning that data must be numerical. Then, it cannot be applied on datasets of categorical or mixed data. Moreover, the number of clusters in the final partition must be specified. As in the (hyper)graph methods (Sec. 2.2.3) a graph partitioning algorithm at the final step must be applied. METIS algorithm is used for this purpose, but in a general way any other graph partitioning algorithm could be applied. The first heuristic discussed in this section (WSPA) has a complexity of $\mathcal{O}(n^2)$ and leads to severe computational limitations. On the other hand, the other heuristic (WBPA) is more efficient with a computational complexity $\mathcal{O}(k\cdot n\cdot m)$.

2.2.9. *Non-negative matrix factorization based methods*

Li *et al.*[58] introduced a clustering ensemble method based on a non-negative matrix factorization process. Nonnegative matrix factorization (NMF)[13] refers to the problem of factorizing a given nonnegative data matrix M into two matrix factors, i.e. $M \approx AB$, while requiring A and B to be non-negative.[58]

In this method, first of all, the following distance between partitions is used

$$\mu(P,P') = \sum_{i,j=1}^{n} \mu_{ij}(P,P') \tag{9}$$

where $\mu_{ij}(P,P')=1$ if x_i and x_j belong to the same cluster in one partition and belong to different clusters in the other, otherwise $\mu_{ij}(P,P')=0$.

Also, the *connectivity matrix* is defined as:

$$M_{ij}(P_v) = \begin{cases} 1, & \exists\, C_t^v \in P_v,\ \text{such that } x_i \in C_t^v \text{ and } x_j \in C_t^v; \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

It is easy to see that $\mu_{ij}(P,P') = |M_{ij}(P) - M_{ij}(P')| = (M_{ij}(P) - M_{ij}(P'))^2$

The consensus partition P^* is defined by the median partition problem using μ as a dissimilarity measure between partitions.

$$P^* = \arg \min_{P\in\mathbb{P}_X} \frac{1}{m} \sum_{v=1}^{m} \mu(P,P_v) = \arg \min_{P\in\mathbb{P}_X} \frac{1}{m} \sum_{v=1}^{m} \sum_{i,j}^{n} (M_{ij}(P) - M_{ij}(P_v))^2$$

Let $U_{ij} = M_{ij}(P^*)$ be the solution to this optimization problem, which is the connectivity matrix of P^*. This optimization problem can take this form:

$$\min_U \sum_{i,j=1}^{n} (\tilde{M}_{ij} - U_{ij})^2 = \min_U \|\tilde{M} - U\|_F^2$$

where $\tilde{M}_{ij} = \frac{1}{m}\sum_{v=1}^{m} M_{ij}(P_v)$ and $\|\cdot\|_F$ denotes the Frobenius norm.

From this moment, several transformations, relaxing some restrictions of the original problem, are made. Finally, the consensus clustering becomes the optimization problem:

$$\min_{Q\geq 0, S\geq 0} \|\tilde{M} - QSQ^T\|^2, \quad s.t.\ Q^TQ = I \tag{11}$$

where the matrix solution U is expressed in terms of the two matrixes Q and S.

The optimization problem (11) can be solved using the following multiplicative update procedure:

$$Q_{ab} \leftarrow Q_{ab}\sqrt{\frac{(\tilde{M}QS)_{ab}}{(QQ^T\tilde{M}QS)_{ab}}} \quad \text{and} \quad S_{bc} \leftarrow S_{bc}\sqrt{\frac{(Q^T\tilde{M}Q)_{bc}}{(Q^TQSQ^TQ)_{bc}}}$$

by this iterative process matrixes Q and S can be obtained, and with these two matrixes, $U = QSQ^T$ is obtained which is the *connectivity matrix* of the consensus partition P^*.

This method defines the consensus clustering as the median partition problem, fixing the distance μ (9) as a measure of likeness between partitions. The original definition of the problem is consecutively relaxed to transform the problem into an optimization problem that can be solved by an iterative process. However, this process can only find local minima, rather than a global minimum of the problem. Although the multiplicative rules are the most common techniques for Non Negative Matrix Factorization, there are other approaches such as Fixed Point Alternating Least Squares algorithms and Quasi-Newton algorithms that can be more efficient and achieve better results than the multiplicative techniques.[14]

2.2.10. *Kernel based methods*

Vega-Pons *et al.*[86] proposed the Weighted Partition Consensus via Kernels (WPCK) algorithm. This algorithm incorporates an intermediate step, called Partition Relevance Analysis (see Sec. 2.3), in the traditional methodology of the clustering ensemble algorithms with the goal of estimating the importance of each partition before the combination process. In this intermediate step, to each partition P_i is assigned a weight value ω_i which represents the relevance of the partition in the cluster ensemble.

In this method, the following similarity measure between partitions is defined, $\tilde{k} : \mathbb{P}_X \times \mathbb{P}_X \to [0,1]$ such that:

$$\tilde{k}(P_i, P_j) = \frac{k(P_i, P_j)}{\sqrt{k(P_i, P_i)k(P_j, P_j)}}$$

where the function $k : \mathbb{P}_X \times \mathbb{P}_X \rightarrow \mathbb{R}_+$ is given by:

$$k(P_i, P_j) = \sum_{S \subseteq X} \delta_S^{P_i} \delta_S^{P_j} \mu(S|P_i)\mu(S|P_j)$$

with

$$\delta_S^P = \begin{cases} 1, & \text{if } \exists C \in P,\ S \subseteq C \\ 0, & \text{otherwise} \end{cases}$$

and $\mu(S|P)$ represents the significance of the subset S in the partition P, which can be computed as $\frac{|S|}{|C|}$ if $\exists C \in P$ such that $S \subseteq C$.

This method defines the consensus partition through the median partition problem by using $\tilde{k}$ as a similarity measure between partitions. Besides, the weights computed in the Partition Relevance Analysis step are taken into account. This way, the consensus partition is defined as:

$$P^* = \arg \max_{P \in \mathbb{P}_X} \sum_{i=1}^{m} \omega_i \cdot \tilde{k}(P, P_i) \tag{12}$$

The function $\tilde{k}$ is proven by Vega-Pons *et al.*[86] to be a *positive semi-definite kernel.*[71] It is known that, if $\tilde{k}$ is a kernel function, there exists a map from $\mathbb{P}_X$ into a Hilbert Space $\mathcal{H}$, $\tilde{\phi} : \mathbb{P}_X \rightarrow \mathcal{H}$ such that $\tilde{k}(P_i, P_j) = \langle \tilde{\phi}(P_i), \tilde{\phi}(P_j) \rangle_{\mathcal{H}}$ (where $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ denotes the dot product in the Hilbert Space $\mathcal{H}$).

The problem (12) is a very difficult optimization problem. However, it can be considered the equivalent problem in the Reproducing Kernel Hilbert Space $\mathcal{H}$ given by:

$$\tilde{\phi}(P^*) = \arg \max_{\tilde{\phi}(P) \in \mathcal{H}} \sum_{i=1}^{m} \omega_i \langle \tilde{\phi}(P), \tilde{\phi}(P_i) \rangle_{\mathcal{H}}$$

In $\mathcal{H}$ this problem can be easily solved. However, to obtain the median partition solving the preimage problem is necessary, i.e. given the solution ψ in $\mathcal{H}$, finding the partition $P^* \in \mathbb{P}_X$ such that $\tilde{\phi}(P^*) = \psi$. The exact solution P^* does not have to exist, since $\mathcal{H}$ is usually larger that $\mathbb{P}_X$. That is why an approximate solution $\hat{P}$ is defined as:

$$\hat{P} = \arg \min_{P \in \mathbb{P}_X} \| \tilde{\phi}(P) - \psi \|_{\mathcal{H}}^2 \tag{13}$$

with

$$\| \tilde{\phi}(P) - \psi \|_{\mathcal{H}}^2 = \tilde{k}(P, P) - 2 \sum_{i=1}^{m} \omega_i \tilde{k}(P, P_i) + \sum_{i=1}^{m} \sum_{j=1}^{m} \omega_i \omega_j \tilde{k}(P_i, P_j) \tag{14}$$

The preimage problem (13) can be solved by using the well-known meta-heuristic simulated annealing.[51] The computational complexity of the WPCK method can be estimated in $\mathcal{O}(n \cdot m \cdot r\text{Max})$, where rMax is the maximum number of iterations used by the simulated annealing as a stop criterion. One of the advantages of this method is that it allows computing which partition in the cluster ensemble is closer to the median partition in a very efficient way.

Following the same idea of using a kernel similarity measure to define the median partition problem another clustering ensemble method called WKF is presented in Ref. 85. In this case, the similarity measure is based on a graph kernel function. A generalization of the WKF denoted by GKWF was presented by Vega-Pons and Ruiz-Shulcloper.[87] This generalization extends the ideas of the WKF method to the categorical and mixed data domains.

This kind of method defines the consensus clustering as the median partition problem using a kernel function as similarity measure between partitions. An exact representation in a Hilbert space of the solution is given, however, to obtain the real P^* the preimage problem (13) must be solved. Its trivial solution is intractable computationally and then, heuristic solutions are proposed. Despite the fact that the simulated annealing works well for this problem, the use of other meta-heuristics could be also analyzed in this case. Besides, if the Partition Relevance Analysis is applied, the quality of the consensus process increases, but the computational cost also increases.

2.2.11. *Fuzzy clustering based methods*

So far, we have exposed the main clustering ensemble methods that accept hard clusterings as input. However, there are some clustering ensemble methods that work with fuzzy clusterings. There are very popular clustering algorithms like EM and fuzzy-c-means[94] that naturally output fuzzy partitions of data. If the results obtained by these methods are *forcibly hardening* i.e. convert fuzzy partition in hard partitions of the data, valuable information for the combination process could be lost. Thus, to combine the fuzzy partitions directly may be more appropriate than hardening first and after that, using a hard clustering ensemble method.[16] The consensus partition obtained by soft clustering ensemble methods could be hard or soft. In this section, we only refer to the methods that output hard final clusterings since they can be used for the same purpose as all previous clustering ensemble methods: given a set of objects, obtaining a hard partitioning of them. The fuzzy clusterings of data are only used in internal steps of the methods.

As in the case of hard clustering ensembles, let $X = \{x_1, x_2, \dots, x_n\}$ be a set of objects, $\mathbb{P} = \{P_1, P_2, \dots, P_m\}$ is a set of partitions of X, where $P_i = \{S_1^i, S_2^i, \dots, S_{d_i}^i\}$ for all $i = 1, 2 \cdots m$. However, in the soft clustering ensemble each S_j^i instead of being a subset of X, can be seen as a function $S_j^i : X \to [0,1]$, where $S_j^i(x_r)$ is the degree of membership of object x_r to the jth cluster of the ith partition.

Among these methods, we can mention sCSPA, sMCLA and sHBGPA[67] which are the fuzzy versions of the algorithms CSPA, MCLA and HGBPA respectively (see Sec. 2.2.3).

sCSPA extends CSPA by changing the way of computing the similarity matrix. Instead of using the co-association matrix as similarity between objects, the matrix

SC is computed as follows: First, each object is viewed as a vector in a $\sum_{i=1}^{m} d_i$ (the number of clusters in $\mathbb{P}$) dimensional space and the Euclidian distance $d_{a,b}$ between the objects x_a and x_b is computed as

$$d_{a,b} = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{d_i} (S_j^i(x_a) - S_j^i(x_b))^2} \tag{15}$$

This can be interpreted as a measure of the difference in the membership of the objects for each cluster. The matrix SC is obtained by converting this distance into a similarity measure where $SC_{a,b} = e^{-d_{a,b}^2}$. After that, the METIS algorithm is used as in the CSPA method, to obtain the final partition.

sMCLA extends MCLA by accepting soft clustering as input. The main difference is the use of the similarity matrix SC that uses Euclidian distance (15) as in the sCSPA method, instead of the similarity between clusters given by the Jaccard index application. Once this meta-graph of clusters is created, the steps of the algorithm are as in CSPA.

Another extension of a hard clustering ensemble algorithm is sHBGF, which is the fuzzy version of HBGF. The HBGF can be trivially adapted to consider soft ensembles since the graph partitioning algorithm METIS accepts weights on the edges of the graph to be partitioned. In sHBGF, the weights (ω) on the edges are set as follows:

- $\omega(i, j) = 0$ if i, j are both clusters or both instances.
- $\omega(i, j) = SC_{i,j}$ otherwise.

Yang *et al.*[96] proposed a modification of the *Evidence Accumulation* clustering ensemble method[30] for the case of fuzzy partition. The *Fuzzy co-association matrix* based on a fuzzy similarity measure is generated to summarize the ensemble of soft partitions. Finally, the traditional techniques in the co-association matrix (see Sec. 2.2.2) based methods can be used to obtain the final partition of the data.

If in the generation step, fuzzy clustering algorithms are available, the use of a soft clustering ensemble algorithm allows making a better modeling of the problem at hand. However, the soft clustering ensemble methods presented in this section are fuzzy versions of (hyper)graph and co-association based methods, and as a consequence, they have similar drawbacks to their hard versions.

2.3. *Clustering discrimination techniques*

In this section, we present some ideas aimed at improving the quality of the combination process. The general methodology in a clustering ensemble algorithm, as we have previously seen, is made up of two steps: Generation and Consensus. Most of the clustering ensemble algorithms use in the consensus step all the partitions obtained in the generation step. Besides, they combine all partitions giving to each one the same significance.

However, in particular situations, all clusterings in the cluster ensemble may not have the same quality, i.e. the information that each one contributes may not be the same. Therefore, a simple average of all clusterings does not have to be the best choice.

In this direction, two different approaches appear. The idea of both approaches is to inspect the generated partitions and make a decision that assists the combination process. The first one consists in selecting a subset of clustering to create an ensemble committee, which will be combined to obtain the final solution. The other approach consists in setting a weight to each partition in order to give a value according to its significance in the clustering ensemble.

Hong *et al.*[42] presented a selective clustering ensemble method that works by evaluating the quality of all obtained clustering results through a resampling technique and selectively choosing part of promising clustering results. For each clustering a fitness value is computed, and partitions are ranked according to this value. Then, partitions with higher fitness values are selected to create the ensemble committee. Also, Fern and Lin[24] introduced an ensemble selection strategy based on the quality and diversity of partitions in the ensemble. As *quality* and *diversity* are concepts not so clearly defined in unsupervised learning, first of all, they explained how to measure the quality and diversity of clustering solutions. After that, they designed three different selection approaches that jointly consider these two concepts.

On the other hand, Vega-Pons *et al.*[85,86] introduced an intermediate step called Partition Relevance Analysis, which assigns a weight to each partition representing how significant each partition is for the combination process. In this step, a set of clustering validity indexes[40] is used. Each index is applied to all partitions in the clustering ensemble. High weights are assigned to partitions with an average behavior with respect to these indexes, and partitions with very different results are considered noise and small weights are associated to them. Also, Li and Ding[59] introduced a way to assign a weight to each clustering before the combination process. In this case, the goal is to reduce the redundancy in the set of partitions. Therefore, if two partitions are very similar, the corresponding weights in the final solution will tend to be small.

These two techniques do not have to be excluding. A selection based on a pondering process or weighting the partitions already selected could be performed. Any of these variants could improve the quality of the final result. However, their use implies an extra computational cost. Therefore, in a practical problem, the users should analyze the peculiarities of the problem at hand, and decide whether to use or not a clustering discrimination technique, according to their requirements.

2.4. *Comparison of the methods*

An experimental comparison of clustering ensemble algorithms was made by Kuncheva *et al.*[54] but only considering co-association and hypergraph based methods.

Bertolacci and Wirth[10] and Goder and Filkov[34] made experimental comparisons of the Mirkin based clustering ensemble algorithms (see Sec. 2.2.4). However, these comparisons are only based on the experimental results obtained by the application of the different methods to a fixed number of datasets. Besides, these comparisons are only among a few number of clustering ensemble methods with similar characteristics. In this sense, we consider that a more complete experimental comparison of clustering ensemble methods is necessary in order to give a benchmark that could be very useful for future publications.

In this section, we make a comparison of the consensus function presented in previous sections taking into account six properties. The idea of this comparison is not to determine which is the *best* consensus function. This is an ambiguous term for this process, which depends on each particular problem, what the users expect in the results and the way to validate the output of the algorithms. The main goal of this comparison is to help the selection of an appropriate kind of consensus function to solve a problem on hand. We explore the general behavior of the different kinds of consensus functions presented in Sec. 2.2. Thus, we unify all the methods based on the same kind of consensus function in one row of Table 1. This way, for each kind of consensus function, we put in Table 1 the general behavior with respect to each one

Table 1. Comparison of consensus functions.

	NCP (1)	DGM (2)	CSO (3)	CPC (4)	TD (5)	CC (6)
Relabeling and Voting	Fixed (*Cumulative Voting*,[7] Variable)	No	No	Yes	Object co-ocurrence	Heuristic dependent
Co-association matrix	Variable	No (*Voting-k-means*,[29] Yes)	No	No	Object co-ocurrence	High
Graph and hypergraph	Variable	No	No	Yes	Object co-ocurrence	Low (*CSPA*,[75] High)
Mirkin distance	Variable	No	No	No	Median partition	Heuristic dependent
Information theory	Variable	No	No	Yes	Object co-ocurrence	Low
Finite mixture models	Variable	No	No	Yes	Object co-ocurrence	Low
Genetic algorithms	Variable	No	No	No	Median partition	Heuristic dependent
LAC algorithms	Variable	Yes	Yes	Yes	Object co-ocurrence	Low
NMF methods	Variable	No	No	No	Median partition	Heuristic dependent
Kernel methods	Variable	No	Yes	No	Median partition	Heuristic dependent
Fuzzy methods	Variable	No	No	Yes	Object co-ocurrence	Low

of the analyzed properties. We highlight in some cells of Table 1 some exceptions that we consider important to take into account. We did not put each particular algorithm in each row of Table 1, because this would lead to a very large table, which would be harder to understand. Besides, we could corroborate that the general case is that all methods of a particular kind of consensus function have the same behavior with respect to each one of the properties analyzed. Exceptions are not very frequent.

We compare the different kinds of consensus functions regarding the following properties:

(1) *Number of Clusters in each Partition* (NCP). This property expresses whether the methods can combine partitions with different number of clusters or not. A method that can combine partitions with a variable number of clusters can be used in a larger number of situations. Demanding that partitions have the same number of clusters is a strong restriction to the clustering ensemble problem.

(2) *Dependency on the Generation Mechanism* (DGM). This characteristic refers to the dependence of the consensus function on a specific type of generation mechanism. A consensus function connected to a fixed generation mechanism could use particular properties of this generation process. However, if the generation mechanism is not appropriate for a particular problem, the results will not be the best. Besides, a consensus function that allows any generation process could be more flexible to different kinds of situations.

(3) *Consider the original Set of Objects* (CSO). Most consensus functions do not consider the original objects and only work with the set of partitions. However, the original objects and their similarity values are additional information that can be useful in the combination. On the other hand, a consensus function that depends on the original objects cannot be applied in situations where the objects are not available in the combination step such as for Knowledge Reuse.[75] Therefore, a consensus function that can make use of the original set of objects if they are available, but it can also work without the original objects, could be desirable.

(4) *The number of clusters in the consensus partition is a parameter of the consensus function* (CPC). A consensus partition capable of determining the optimum number of cluster in the consensus partition is generally preferable. However, if in a particular problem the users know how many clusters they want in the consensus clustering, a consensus function where the number of clusters must be specified, should be more appropriate. However, the algorithms that can work without the specification of the number of cluster can be easily transformed to make use of the number of clusters as a parameter and to restrict the solution to this parameter. On the other hand, the methods that need the specification of the number of clusters in the consensus partition usually cannot be easily transformed to work independently from this parameter. Thus, the consensus function techniques that can work without the number of clusters specification are more flexible than the ones which require the number of cluster specification.

(5) *Theoretical Definition* (TD). Consensus functions can be based on two approaches *objects co-occurrence* or *the median partition*, (see Sec. 2.2). Consensus methods that face the problem through the search of the median partitions are theoretically better defined. However, in practice, they are heuristics to solve a hard optimization problem, therefore, the theoretical strength of these methods depend on the particular heuristics.

(6) *Computational Complexity* (CC). In the case of the computational complexity, we use three values (*low*, *high* and *heuristic dependent*) because of the following reasons. In each kind of consensus function there could be different methods with different computational complexity. For all clustering ensemble methods, their exact computational complexity is not given in terms of the same variables. The exact computational complexity of all clustering ensemble methods is not easy to estimate. This is because some of them are heuristics and it is not easy to determine how many steps are needed to reach a convergence criterion. Besides, clustering ensemble algorithms with a quadratic or superior computational complexity in the number of objects cannot be applied to large datasets. Thus, we used the quadratic cost on the number of objects as a threshold to determine whether an algorithm has *high* or *low* computational complexity. We use the value *Heuristic dependent* when it is very difficult to determine the computational complexity because it depends on the heuristic applied, on the particular problem and on the convergence criteria used in each case. However, details about the exact computational complexity of some algorithms can be found in the sections where they were presented.

3. Applications

The recent progress in clustering ensemble techniques is to a big extent endorsed by its application to several fields of investigation. There is a large variety of problems in which the clustering ensemble algorithms can be applied. In principle, as clustering ensembles try to improve the quality of clustering results, they can be directly used in almost all cluster analysis problems, e.g. image segmentation, bioinformatics, document retrieval and data mining.[47]

In particular, Gionis *et al.*[32] showed how clustering ensemble algorithms can be useful for improving the clustering robustness, clustering categorical data and heterogeneous data, identifying the correct number of clusters and detecting outliers.

Besides, there are some papers about direct applications of clustering ensembles to some research fields such as image segmentation: Wattuya *et al.*,[91] Wattuya and Jiang,[92] Forestier *et al.*,[27] Yu *et al.*,[99] Zhang *et al.*,[101] Singh *et al.*[74] and Chang *et al.*[12]; document clustering: Greene and Cunningham,[38] Shinnou and Sasaki,[73] Gonzalez and Turno,[35] and Xu *et al.*[95]; feature extraction: Hong *et al.*,[43] and Hong *et al.*[44]; bioinformatics, in particular in gene expression analysis: Filkov and Skiena,[25] Deodhar and Ghosh,[17] Avogadri and Valentini,[5] Yu and Wong,[100] Hu *et al.*,[45] and Kashef and

Kamel[50]; physics problems: Wouterse and Philipse[93]; medical applications: Shen *et al.*[72]; among others.

Clustering ensemble methods developed for a specific application purpose should take into account the peculiarities of the problem at hand. The kind of clustering ensemble algorithm should be selected according to the specific requirements of each application. For instance, in image segmentation problems, graph representation of images are very convenient since neighboring relations among pixels can easily be taken into account by this structure. Besides, in image segmentation the computational cost is an important issue because images usually have a large number of pixels. Hence, graph based clustering ensemble methods could be an appropriate choice in the segmentation ensemble context.

4. Conclusions

Clustering ensemble has become a leading technique when facing cluster analysis problems, due to its capacity for improving the results of simple clustering algorithms. The combination process integrates information from all partitions in the ensemble, where possible errors in simple clustering algorithms could be compensated. That way, the consensus clustering, obtained from a set of clusterings of the same dataset, represents an appropriate solution.

In this paper, we explore the main clustering ensemble approaches taking into account their theoretical definition as well as the mathematical and computational tools used by each method. The bibliographical compilation is presented by using a unified notation, through a simple and homogeneous exposition of their fundamentals. That way, this survey makes the understanding of the methods developed until now easier to the reader.

Due to the unsupervised nature of these techniques, it is not adequate to talk about the *best* clustering ensemble method. Nevertheless, we can still establish a comparison among these methods and determine, for specific conditions, which one may be the most appropriate. We made a critical analysis and comparison of the methods, taking into account different parameters. The main advantages and disadvantages of each method can be helpful to the users to select the convenient method to solve their particular problem.

From the analysis of the different techniques, we consider that in future works the following issues should be tackled. It is necessary to create clustering ensemble methods well-founded from the theoretical point of view. The combination process should be supported by a theoretical analysis that endorses the use of the method. A simple average of all partitions in the ensemble does not have to be the best consensus. By making a differentiation process among the partitions in the ensemble (such as in Sec. 2.3) the quality of the consensus clustering could be improved. The existing clustering ensemble methods are addressed to combine partitions. However, some clustering algorithms give covers (overlapped clusters) as output, instead of partitions of the data. Therefore, combination of covers could improve the quality of this kind of clustering algorithms. Most of the clustering ensemble algorithms only use the set of partitions in the combination process. However, the original set of objects and their similarity values represent extra information that can be useful for the combination, but generally it is not used. In order to deal with large datasets, efficient clustering ensemble algorithms with a computational complexity lower than $\mathcal{O}(n^2)$, are necessary. Finally, considering the number of papers already published on this topic and the current activity in this investigation field, we think that a general experimental evaluation of the existing methods is necessary. Such benchmark could be very useful for the comparison and evaluation of the existing and future clustering ensemble methods.

References

1. D. Abdala, P. Wattuya and X. Jiang, Ensemble clustering via random walker consensus strategy, *20th Int. Conf. Pattern Recognition*, ICPR 2010, pp. 1433–1436.
2. N. Alion, M. Charikar and A. Newman, Aggregating inconsistent information: ranking and clustering, *37th Symp. Theory of Computing (STOC)* (2005), pp. 684–693.
3. E. Amigó, J. Gonzalo, J. Artiles and F. Verdejo, A comparison of extrinsic clustering evaluation metrics based on formal constraints, *Inf. Retr.* **12**(4) (2009) 461–486.
4. M. Analoui and N. Sadighian, Solving cluster ensemble problems by correlation's matrix & GA, *IFIP Int. Fed. Inform. Process.* **228** (2006) 227–231.
5. R. Avogadri and G. Valentini, Ensemble clustering with a fuzzy approach, *Studies in Comput. Intell.* (SCI) **126** (2008) 49–69.
6. H. G. Ayad and M. S. Kamel, On voting-based consensus of cluster ensembles, *Patt. Recogn.* **43** (2010) 1943–1953.
7. H. G. Ayad and M. S. Kamel, Cumulative voting consensus method for partitions with a variable number of clusters, *IEEE Trans. Patt. Anal. Mach. Intell.* **30**(1) (2008) 160–173.
8. J. Bakus, M. F. Hussin and M. Kamel, A SOM-based document clustering using phrases, *Proc. 9th Int. Conf. Neural Information Procesing* (ICONIP02) (2002), pp. 2212–2216.
9. A. Ben-Hur, A. Elisseeff and I. Guyon, A stability based method for discovering structure in clustered data, *Pacific Symp. Biocomputing* (2002), pp. 6–17.
10. M. Bertolacci and A. Wirth, Are approximation algorithms for consensus clustering worthwhile?, *Proc. Seventh SIAM ICDM* (2007), pp. 437–442.
11. M. Brun, C. Sima, J. Hua, J. Lowey, B. Carroll, E. Suh and E. R. Dougherty, Model-based evaluation of clustering validation measures, *Patt. Recogn.* **40** (2007) 807–824.
12. Y. Chang, D.-J. Lee, Y. Hong, J. Archibald and D. Liang, A robust color image quantization algorithm based on knowledge reuse of *k*-means clustering ensemble, *J. Multimed.* **3** (2008) 20–27.
13. A. Cichocki, M. Mrup, P. Smaragdis, W. Wang and R. Zdunek, *Advances in Nonnegative Matrix and Tensor Factorization, Computational Intelligence & Neuroscience* (Hindawi Publishing Corporation, 2008).
14. A. Cichocki, R. Zdunek and S. Amari, Nonnegative matrix and tensor factorization, *IEEE Sign. Process. Mag.* **142** (2008) 142–145.
15. T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd edn., Wiley Series in Telecommunications and Signal Processing (Wiley-Interscience, 2006).
16. J. V. de Olivera and W. Pedrycz, *Advances in Fuzzy Clustering and Its Applications* (Wiley, 2007).

17. M. Deodhar and J. Ghosh, Consensus clustering for detection of overlapping clusters in microarray data, *Data Mining Workshops. ICDM Workshops 2006* (2006), pp. 104–108.
18. E. Dimitriadou, A. Weingessel and K. Hornik, An ensemble method for clustering, *ICANN* (2001), pp. 217–224.
19. E. Dimitriadou, A. Weingessel and K. Hornik, A ccombination scheme for fuzzy clustering, *Int. J. Patt. Recogn. Artif. Intell.* **16**(7) (2002) 901–912.
20. C. Domeniconi and M. Al-Razgan, Weighted cluster ensembles: methods and analysis, *ACM Trans. Knowl. Discov. Data* **2**(4) (2009) 1–40.
21. C. Domeniconi, D. Gunopulos, S. Ma, B. Yan, M. Al-Razgan and D. Papadopoulos, Locally adaptive metrics for clustering high dimensionl data, *Data Min. Knowled. Discov. J.* **14**(1) (2007) 63–97.
22. S. Dudoit and J. Fridlyand, Bagging to improve the accuracy of a clustering procedure, *Bioinformatics* **19**(9) (2003) 1090–1099.
23. X. Z. Fern and C. E. Brodley, Solving cluster ensemble problems by bipartite graph partitioning, *ICML'04: Proc. Twenty-First Int. Conf. Machine Learning* (ACM, New York, NY, USA, 2004), p. 36.
24. X. Z. Fern and W. Lin, Cluster ensemble selection, *Stat. Anal. Data Min.* **1**(3) (2008) 128–141.
25. V. Filkov and S. Skiena, Integrating microarray data by consensus clustering, *Int. J. Artif. Intell. Tools* **13**(4) (2004) 863–880.
26. B. Fischer and J. Buhmann, Bagging for path-based clustering, *IEEE Trans. Patt. Anal. Mach. Intell.* **25**(11) (2003) 1411–1415.
27. G. Forestier, C. Wemmert and P. Ganarski, Collaborative multi-strategical clustering for object-oriented image analysis, *Studies in Computational Intelligence*, Vol. 126 (Springer Berlin/Heidelberg, 2008), pp. 71–88.
28. E. B. Fowlkes and J. A. Mallows, A method for comparing two hierarchical clusterings, *J. Amer. Statist. Assoc.* **78**(383) (1983) 553–569.
29. A. Fred, Finding consistent clusters in data partitions, *3rd. Int. Workshop on Multiple Classifier Systems* (2001), pp. 309–318.
30. A. L. N. Fred and A. K. Jain, Combining multiple clustering using evidence accumulation, *IEEE Trans. Patt. Anal. Mach. Intell.* **27** (2005) 835–850.
31. R. Ghaemi, M. N. Sulaiman, H. Ibrahim and N. Mustapha, A survey: clustering ensembles thechniques, *Proc. World Acad. Sci. Engin. Technol.* **38** (2009) 644–653.
32. A. Gionis, H. Mannila and P. Tsaparas, Clustering aggregation, *ACM Trans. Knowl. Discov. Data* **1**(1) (2007) 341–352.
33. M. Gluck and J. Corter, Information, uncertainty, and the utility of categories, *Proc. Seventh Annual Conf. Cognitive Science Society* (Lawrence Erlbaum: Hillsdale, NJ, 1985), pp. 283–287.
34. A. Goder and V. Filkov, *Consensus Clustering Algorithms: Comparison and Refinement*, eds. J. I. Munro and D. Wagner (ALENEX, SIAM, 2008), pp. 109–117.
35. E. Gonzalez and J. Turmo, Comparing non-parametric ensemble methods for document clustering, *Natural Language and Information Systems*, LNCS, Vol. 5039 (2008), pp. 245–256.
36. A. Gordon and M. Vichi, Fuzzy partition models for fitting a set of partitions, *Psychometrika* **66**(2) (2001) 229–248.
37. L. Grady, Random walks for image segmentation, *IEEE Trans. Patt. Anal. Mach. Intell.* **28**(11) (2006) 1768–1783.
38. D. Greene and P. Cunningham, Efficient ensemble method for document clustering, Tech. Rep., Department of Computer Science, Trinity College Dublin (2006).

39. M. Grotschel and Y. Wakabayashi, A cutting plane algorithm for a clustering problem, *Math. Program.* **45** (1989) 59−96.
40. M. Halkidi, Y. Batistakis and M. Vazirgiannis, On clustering validation techniques, *Intell. Inf. Syst. J.* **17** (2001) 107−145.
41. J. Handl, J. Knowles and D. Kell, Computational cluster validation in post-genomic data analysis, *Bioinformatics* **21** (2005) 3201−3212.
42. Y. Hong, S. Kwong, H. Wang and Q. Ren, Resampling-based selective clustering ensembles, *Patt. Recogn. Lett.* **30**(3) (2009) 298−305.
43. Y. Hong, S. Kwong, Y. Chang and Q. Ren, Unsupervised feature selection using clustering ensembles and population based incremental learning algorithm, *Patt. Recogn.* **41** (2008) 2742−2756.
44. Y. Hong, S. Kwong, Y. Chang and Q. Ren, Consensus unsupervised feature ranking from multiple views, *Patt. Recogn. Lett.* **29** (2008) 595−602.
45. X. Hu, E. Park and X. Zhang, Microarray gene cluster identification and annotation through cluster ensemble and EM-based informative textual summarization, *IEEE Trans. Inf. Technol. Biomed.* **13**(5) (2009) 832−840.
46. N. Iam-on, T. Boongoen and S. Garrett, Refining pairwise similarity matrix for cluster ensemble problem with cluster relations, eds. J. F. Boulicaut, M. R. Berthold and T. Horvath, DS 2008, LNAI, Vol. 5255 (Springer-Verlag Berlin Heidelberg, 2008), pp. 222−233.
47. A. K. Jain, M. Murty and P. Flynn, Data clustering: A review, *ACM Comput. Surv. (CSUR)* **31**(3) (1999) 264−323.
48. G. Karypis, R. Aggarwal, V. Kumar and S. Shekhar, Multilevel hypergraph partitioning: application in VLSI domain, *DAC'97: Proc. 34th Ann. Conf. Design Automation* (ACM, New York, NY, USA, 1997), pp. 526−529.
49. G. Karypis and V. Kumar, A fast and high quality multilevel scheme for partitioning irregular graphs, *SIAM J. Scient. Comput.* **20** (1998) 359−392.
50. R. Kashef and M. Kamel, Cooperative partitional-divisive clustering and its application in gene expression analysis, *7th IEEE Int. Conf. Bioinformatics and Bioengineering, BIBE* (2007), pp. 116−122.
51. S. Kirkpatrick, C. Gellat and M. Vecchi, Optimization by simulated annealing, *Science* **220** (1983) 671−680.
52. M. Krivanek and J. Moravek, Hard problems in hierarchical-tree clustering, *Acta Inform.* **3** (1998) 311−323.
53. H. Kuhn, The Hungarian method for the assignment problem, *Naval Res. Logistic Quart.* **2** (1955) 83−97.
54. L. I. Kuncheva, S. T. Hadjitoorov and L. P. Todorova, Experimental comparison of cluster ensemble methods, *9th Int. Conf. Information Fusion* (2006), pp. 1−7.
55. L. Lam and C. Y. Suen, Application of majority voting to pattern recognition: an analysis of its behavior and performance, *IEEE Trans. Syst. Man Cybern.* **27**(5) (1997) 553−568.
56. T. Li, M. Ogihara and S. Ma, On combining multiple clusterings: an overview and a new perspective, *Appl. Intell.* doi:10.1007/s10489-009-0160-4.
57. Y. Li, J. Yu, P. Hao and Z. Li, Clustering ensembles based on normalized edges, eds. Z. H. Zhou, H. Li and Q. Yang, *PAKDD*, LNAI, Vol. 4426 (Springer-Verlag Berlin Heidelberg, 2007), pp. 664−671.
58. T. Li, C. Ding and M. I. Jordan, Solving consensus and semi-supervised clustering problems using nonnegative matrix factorization, *ICDM '07: Proc. 2007 Seventh IEEE Int. Conf. Data Mining* (IEEE Computer Society, Washington, DC, USA, 2007), pp. 577−582.

59. T. Li and C. Ding, Weighted consensus clustering, *SIAM Int. Conf. Data Mining (SDM)* (2008), pp. 798–809.

60. H. Luo, F. Jing and X. Xie, Combining multiple clusterings using information theory based genetic algorithm, *IEEE Int. Conf. Computational Intelligence and Security* **1** (2006) 84–89.

61. G. McLachlan and D. Peel, *Finite Mixture Models* (John Wiley & Sons, New York, 2000).

62. M. Meilă, Comparing clusterings—an information based distance, *J. Multiv. Anal.* **98**(5) (2007) 873–895.

63. B. G. Mirkin, *Mathematical Classification and Clustering* (Kluwer Academic Press, Dordrecht, 1996).

64. B. Mirkin, Reinterpreting the category utility function, *Mach. Learn.* **45**(2) (2001) 219–228.

65. A. Y. Ng, M. I. Jordan and Y. Weiss, On spectral clustering: Analysis and an algorithm, *Advances in Neural Information Processing Systems*, Vol. 14 (MIT Press, 2001), pp. 849–856.

66. D. Pfitzner, R. Leibbrandt and D. Powers, Characterization and evaluation of similarity measures for pairs of clusterings, *Knowl. Inf. Syst.* **19** (2009) 361–394.

67. K. Punera and J. Ghosh, Consensus-based ensembles of soft clusterings, *Appl. Artif. Intell.* **22**(7&8) (2008) 780–810.

68. W. M. Rand, Objective criteria for the evaluation of clustering methods, *J. Amer. Statist. Assoc.* **66** (1971) 846–850.

69. S. Régnier, Sur quelques aspects mathematiques des problemes de classification automatique., *ICC Bull.* **4** (1965) 175–191.

70. A. Rosenberg and J. Hirschberg, V-measure: A conditional entropy-based external cluster evaluation measure, *Proc. 2007 Joint Conf. Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)* (2007), pp. 410–420.

71. B. Scholkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond* (MIT Press, Cambridge, MA, USA, 2002).

72. J. Shen, P. Lee, J. Holden and H. Shatkay, Using cluster ensemble and validation to identify subtypes of paervasive developmental disorders, *Proc. AMIA Symp.* (Chicago, 2007), pp. 666–670.

73. H. Shinnou and M. Sasaki, Ensemble document clustering using weighted hypergraph generated by NMF, *ACL '07: Proc. 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, Association for Computational Linguistics, Morristown, NJ, USA, 2007, pp. 77–80.

74. J. P. J. X. V. Singh and L. Mukerjee, Ensemble clustering using semidefinite programming, *Advance in Neural Information Processing Systems* **20** (2007) 1353–1360.

75. A. Strehl and J. Ghosh, Cluster ensembles: A knowledge reuse framework for combining multiple partitions, *J. Mach. Learn. Res.* **3** (2002) 583–617.

76. A. Topchy, A. K. Jain and W. Punch, A mixture model of clustering ensembles, *SIAM Int. Conf. Data Mining* (2004), pp. 379–390.

77. A. Topchy, A. K. Jain and W. Punch, Combining multiple weak clusterings, *ICDM'03: Proc. Third IEEE Int. Conf. Data Mining* (IEEE Computer Society, Washington, DC, USA, 2003), pp. 331–338.

78. A. P. Topchy, A. K. Jain and W. F. Punch, Clustering ensembles: Models of consensus and weak partitions, *IEEE Trans. Patt. Anal. Mach. Intell.* **27**(12) (2005) 1866–1881.

79. A. Topchy, M. Law, A. Jain and A. Fred, Analysis of consensus partition in cluster ensemble, *Int. Conf. Data Mining* (2004), pp. 225–232.

80. K. Tumer and A. Agogino, Ensemble clustering with voting active clusters, *Patt. Recogn. Lett.* **29** (2008) 1947–1953.
81. S. Van Dongen, Performance criteria for graph clustering and Markov cluster experiments, Tech. Rep. INS-R0012, Centre for Mathematics and Computer Science (2000).
82. C. Van Rijsbergen, Foundation of evaluation, *J. Document.* **30**(4) (1974) 365–373.
83. A. van Zuylen, Deterministic approximation algorithm for ranking and clustering problems, Tech. Rep. 1431, OIRIE, Cornell University (2005).
84. V. V. Vazirani, *Approximation Algorithms* (Springer-Verlag, 2001).
85. S. Vega-Pons, J. Correa-Morris and J. Ruiz-Shulcloper, Weighted cluster ensemble using a kernel consensus function, *CIARP '08: 13th Iberoamerican Congress on Pattern Recognition*, LNCS, Vol. 5197 (Springer-Verlag, Berlin, Heidelberg, 2008), pp. 195–202.
86. S. Vega-Pons, J. Correa-Morris and J. Ruiz-Shulcloper, Weighted partition consensus via kernels, *Patt. Recogn.* **43**(8) (2010) 2712–2724.
87. S. Vega-Pons and J. Ruiz-Shulcloper, Clustering ensemble method for heterogeneous partitions, eds. E. Bayro-Corrochano and J.-O. Eklundh, CIARP 2009, LNCS, 2009, Vol. 5856, pp. 481–488.
88. Y. Wakabayashi, Aggregation of binary relations: Algorithmic and polyhedral investigations, Ph.D. thesis, Universitat Augsburg (1986).
89. Y. Wakabayashi, The complexity of computing median of relations, *Resenhas IME-USP* **3** (1998) 311–323.
90. X. Wang, C. Yang and J. Zhou, Clustering aggregation by probability accumulation, *Patt. Recogn.* **42**(5) (2009) 668–675.
91. P. Wattuya, K. Rothaus, J. S. Prassni and X. Jiang, A random walker based approach to combining multiple segmentations, *19th Int. Conf. Pattern Recognition, ICPR* 2008, pp. 1–4.
92. P. Wattuya and X. Jiang, Ensemble combination for solving the parameter selection problem in image segmentation, *Structural, Syntactic, and Statistical Pattern Recognition*, LNCS, Vol. 5342 (Springer Berlin/Heidelberg, 2008), pp. 392–401.
93. A. Wouterse and A. P. Philipse, Geometrical cluster ensemble analysis of random sphere packings, *J. Chem. Phys.* **125** (2006) 194709.1–194709.10.
94. R. Xu and D. Wunsch II, Survey of clustering algorithms, *IEEE Trans. Neural Networks* **16** (2005) 645–678.
95. S. Xu, Z. Lu and G. Gu, An efficient spectral method for document cluster ensemble, *9th Int. Conf. Young Computer Scientists* (2008), pp. 808–813.
96. L. Yang, H. Lv and W. Wang, Soft cluster ensemble based on fuzzy similarity measure, *IMACS Multiconf. Computational Engineering in Systems Applications*, Vol. 2 (2006), pp. 1994–1997.
97. H.-S. Yoon, S.-Y. Ahn, S.-H. Lee, S.-B. Cho and J. H. Kim, Heterogeneous clustering ensemble method for combining different cluster results, *BioDM 2006*, LNBI, Vol. 3916 (2006), pp. 82–92.
98. H.-S. Yoon, S.-H. Lee, S.-B. Cho and J. H. Kim, A novel framework for discovering robust cluster results, *DS 2006*, LNAI, Vol. 4265 (Springer-Verlag Berlin Heidelberg, 2006), pp. 373–377.
99. Z. Yu, S. Zhang, H.-S. Wong and J. Zhang, Image segmentation based on cluster ensemble, *Advances in Neural Networks*, LNCS Vol. 4493 (Springer Berlin/Heidelberg, 2007), pp. 894–903.
100. Z. Yu and H. Wong, Class discovery from gene expression data based on perturbation and cluster ensemble, *IEEE Trans. Nanobiosci.* **8**(2) (2009) 147–160.
101. X. Zhang, L. Jiao, F. Liu, L. Bo and M. Gong, Spectral clustering ensemble applied to SAR image segmentation, *IEEE Trans. Geosci. Rem. Sens.* **46**(7) (2008) 2126–2136.

102. Y. Zhao and G. Karypis, Criterion functions for document clustering: Experiments and analysis, Tech. Rep. TR 01-40, Department of Computer Science, University of Minnesota, Minneapolis, MN (2001).

103. Z.-H. Zhou and W. Tang, Clusterer ensemble, *Knowledge-Based Syst.* **19** (2006) 77–83.

Sandro Vega-Pons graduated in computer science from the University of Havana in 2007 and received his Ph.D. degree from the same institution in 2011. He is currently working as a researcher in the Advanced Technologies Application Center, Cuba.

His research interests mainly include pattern recognition, cluster analysis, kernel methods and image processing.

José Ruiz-Shulcloper graduated from the University of Havana in 1972, received his Ph.D. degree in mathematics from Moscow State University in 1978 and Dr. in Sciences from Central University of Las Villas, Cuba in 2010. Since 1978 he is the reader in the research area of logical combinatorial pattern recognition and author of more than 100 scientific publications on pattern recognition and data mining.