Multiple Consensuses Clustering by Iterative Merging/Splitting of Clustering Patterns

Atheer Al-najdi(✉), Nicolas Pasquier, and Frédéric Precioso

Univ. Nice Sophia Antipolis, CNRS, I3S, UMR 7271, 06900 Sophia Antipolis, France
{alnajdi,pasquier,precioso}@i3s.unice.fr

Abstract. The existence of many clustering algorithms with variable performance on each dataset made the clustering task difficult. Consensus clustering tries to solve this problem by combining the partitions generated by different algorithms to build a new solution that is more stable and achieves better results. In this work, we propose a new consensus method that, unlike others, give more insight on the relations between the different partitions in the clusterings ensemble, by using the frequent closed itemsets technique, usually used for association rules discovery. Instead of generating one consensus, our method generates multiple consensuses based on varying the number of base clusterings, and links these solutions in a hierarchical representation that eases the selection of the best clustering. This hierarchical view also provides an analysis tool, for example to discover strong clusters or outlier instances.

Keywords: Unsupervised learning · Clustering · Consensus clustering · Ensemble clustering · Frequent closed itemsets

1 Introduction

Although the last decades witnessed the development of many clustering algorithms, getting a "good" quality partitioning remains a difficult task. This problem has many dimensions, one of them is the fact that the results of clustering algorithms are data-dependent. An algorithm can achieve good results in some datasets while in others it does not. This is because each is designed to discover a specific clustering structure in the dataset. Another aspect of the problem is the effect of algorithm parameters on the results since changing the settings may produce different partitioning in terms of the number and size of clusters. Defining what should be a "correct" (or a "good") clustering also contributes to the problem, despite the existence of many validation measures whether internal or external.[1] External validation measures are not always applicable, because usually class labels are not provided, especially for large datasets. Moreover, Färber *et al.* [6] states that using such measures, usually applied to synthetic datasets, may not be sound for real datasets because the classes may contain

[1] More details about validation measures can be found in [4], [12] and [19].

P. Perner (Ed.): MLDM 2016, LNAI 9729, pp. 790–804, 2016.
DOI: 10.1007/978-3-319-41920-6_60

internal structures that the present attributes may not allow to retrieve, or also the classes may contain anomalies. On the other hand, internal validation measures may overrate the results of a clustering algorithm which targets the same underlying structure model as the one targeted by the measure.

From many available clustering algorithms with variable outcome, researchers focused recently on the possibility of combining multiple clusterings, called *base clusterings*, to build a new consensus solution that can be better than what each single base clustering could achieve. Such process is called *consensus clustering* or *aggregation of clusterings*. It involves 2 steps: first, building an ensemble of partitions (i.e. the combination of all the partitions provided by the base clustering algorithms), then applying a consensus function. Some consensus clustering approaches impose limitations on the ensemble, such as all the base clusterings must produce the same number of clusters. Other approaches require consensus function with high storage or time complexities. In this work, we present a new category of consensus clustering methods, that is, a pattern-based consensus generation using the frequent closed itemset technique from the frequent pattern mining domain. Since clustering quality depends more on their meaningfulness to the analyst, our method involves generating multiple consensuses by varying the number of base clusterings. The results are presented in a tree of consensuses that not just facilitates the selection of the preferred partitioning, but also depicts how the clusters are generated, and what clusters are more stable than others. We also present a recommended solution, which is the consensus that is the most similar to the ensemble.

In the next section, we summarize some of the consensus clustering methods. Section 3 details the proposed approach. Experimental results are shown in Sect. 4. We conclude in Sect. 5.

2 Related Work

Consensus clustering methods can be organized into several categories according to the underlying approach [10],[24]:

- **Graph partitioning methods:** The problem of consensus clustering is formulated as a graph partitioning problem where the instances and clusters of the base clusterings are used to build the vertices and edges of the graph respectively. Examples of such consensus methods: Cluster-based Similarity Partitioning Algorithm (CSPA), HyperGraph Partitioning Algorithm (HGPA), Meta CLustering Algorithm (MCLA) (Strehl and Ghosh [21]), and Hybrid Bipartite Graph Formulation (HBGF) (Fern and Brodley [7]).
- **Voting methods:** The objective is to match the cluster labels in all base partitions, then perform a voting procedure to find the final grouping of the instances. One limitation in voting methods is that they require the clusterings in the ensemble to produce the same number of clusters as the targeted consensus. Example: the Plurality Voting method (Dudoit and Fridlyand [5], Fischer and Buhmann [8]).

- **Co-association based methods:** A co-association matrix can be used to record how many times 2 instances belong to the same cluster in the ensemble of all partitions. Thus, the matrix defines a new measure of similarity between the instances. As an example, Fred and Jain [9] applied a minimum spanning tree algorithm on the matrix to generate a consensus clustering.
- **Information based methods:** The consensus function here tries to find a clustering that shares the maximum information with the ensemble. Topchy et. al. [22] proposed to use the Category Utility function in order to define the similarity between partitions. The resulting consensus represents the "median" partition which is the most similar to the ensemble.

More details can be found in the surveys by Ghaemi *et al.* [10], Sarumathi *et al.* [20], and Vega-Pons & Ruiz-Shulcloper [24].

3 Pattern-Based Consensus Generation

Our approach is based on discovering *clustering patterns* among the ensemble. Pattern mining and association rule discovery aims at identifying relationships between items in very large datasets [17]. If we consider each cluster in the ensemble as an item, then using pattern mining enables us to discover relationships between the clusters, by identifying the sets of instances that are clustered together by sets of base clusterings.

One of the powerful techniques is the *Frequent Closed Itemset* (FCI) [17]. Its objective is to find the maximal sets of items (clusters) that are common to sets of objects (instances). FCI generates patterns of fewer items only if they become more frequent than their maximal sets, thus eliminating many redundant clustering patterns for our approach, and reducing memory consumption and execution times compared to the approaches based on generating all frequent itemsets.[2]

Our consensus generation process starts by creating a clustering ensemble, from which a binary membership matrix is built. FCI technique is applied on the binary matrix to find clustering patterns, and finally we apply our proposed algorithm to generate the consensus partition, as explained in the following subsections.

3.1 Clustering Ensemble

We do not impose any restriction on the selection of the base clustering algorithms or their settings, as long as they produce hard partitions, that is, each instance belongs to only one cluster. However, changing the base algorithms (partition-based, hierarchical, density-based, etc) and/or theirs settings is preferable to ensure the diversity in the ensemble, which makes the consensus more powerful as it can benefit from the different shapes and sizes of base clusters.

[2] See [2] for an extensive survey on association rule mining.

3.2 Cluster Membership Matrix

After generating a clustering ensemble by partitioning the dataset considering different clustering algorithms, a cluster membership matrix $\mathcal{M}$ is built. $\mathcal{M}$ is used in several consensus clustering methods, as in [1] and [21]. It consists of n rows and m columns, where n is the number of instances, and m is the total number of clusters of all base clusterings. The membership matrix records the binary relation between instances and clusters as given in definition 1.

Definition 1. *A cluster membership matrix $\mathcal{M}$ is a triplet $(\mathcal{I}, \mathcal{C}, \mathcal{R})$ where $\mathcal{I}$ is a finite set of instances represented as rows, $\mathcal{C}$ is a finite set of clusters represented as columns, and $\mathcal{R}$ is a binary relation defining relationships between rows and columns: $\mathcal{R} \subseteq \mathcal{I} \times \mathcal{C}$. Every couple $(i, c) \in \mathcal{R}$, where $i \in \mathcal{I}$ and $c \in \mathcal{C}$, means that instance i belongs to cluster c. This binary relation is represented in the matrix by 1 at $\mathcal{M}_{ic}$, and 0 if there is no relation.*

Let us take as an illustrative example a dataset of nine instances $\mathcal{D} = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Suppose that we used 5 base clustering algorithms to produce 5 partitions of $\mathcal{D}$ as follows: $P1 = \{\{1, 2, 3, 4\}, \{5, 6, 7, 8, 9\}\}$, $P2 = \{\{1, 2, 3\}, \{4, 5, 6, 7, 8, 9\}\}$, $P3 = \{\{1, 2, 3, 4, 5\}, \{6, 7, 8, 9\}\}$, $P4 = \{\{6, 7\}, \{1, 2, 3, 4, 5, 8, 9\}\}$, and $P5 = \{\{4, 5, 6, 7, 8, 9\}, \{1, 2\}\}$. Table 1 presents the membership matrix. Each column P^i_j represents cluster j in partition i as a binary vector where values '1' identify the instances that belong to the cluster. In pattern mining domain, each column in $\mathcal{M}$ represents an item, as defined below.

Table 1. Example cluster membership matrix.

Instance ID	P^1_1	P^1_2	P^2_1	P^2_2	P^3_1	P^3_2	P^4_1	P^4_2	P^5_1	P^5_2
1	1	0	1	0	1	0	0	1	0	1
2	1	0	1	0	1	0	0	1	0	1
3	1	0	1	0	1	0	0	1	1	0
4	1	0	0	1	1	0	0	1	1	0
5	0	1	0	1	1	0	0	1	1	0
6	0	1	0	1	0	1	1	0	1	0
7	0	1	0	1	0	1	1	0	1	0
8	0	1	0	1	0	1	0	1	1	0
9	0	1	0	1	0	1	0	1	1	0

Definition 2. *An item of a cluster membership matrix $\mathcal{M} = (\mathcal{I}, \mathcal{C}, \mathcal{R})$ is a cluster identifier $c \in \mathcal{C}$.*
An itemset is a non-empty finite set of items $C = \{c_1, ..., c_n\} \subseteq \mathcal{C}$ in $\mathcal{M}$.
An itemset $C \subseteq \mathcal{C}$ is frequent in $\mathcal{M}$ if and only if its frequency, called support, in $\mathcal{M}$ defined as $support(C) = |\{I \in \mathcal{I} \mid \forall i \in I, \forall c \in C,$ we have $(i, c) \in \mathcal{R}\}|$ is greater than or equal to the user-defined minsupport threshold.

3.3 Clustering Patterns

The rows in $\mathcal{M}$ present binary patterns (frequent itemsets) that tell the clusters to which each instance belongs. Using FCI methodology, we can find all the sets of instances that share the same clustering pattern. FCI discovers patterns from not only the full set of base clusterings, but also from their subsets, as long as they satisfy the FCI properties defined in [17]. We call the combination of the set of instance identifiers with its corresponding set of cluster identifiers a *Frequent Closed Pattern* (FCP). Table 2 shows the set of the FCPs extracted from Table 1.[3]

Definition 3. *A Frequent Closed Pattern $P = (C, I)$ in the cluster membership matrix $\mathcal{M} = (\mathcal{I}, \mathcal{C}, \mathcal{R})$ is a pair of sets $C \subset \mathcal{C}$ and $I \subset \mathcal{I}$ such that:*
i) $\forall i \in I$ and $\forall c \in C$, we have $(i, c) \in \mathcal{R}$.
ii) $|I| \geq minsupport$, i.e., C is a frequent itemset.
iii) $\nexists i' \in \mathcal{I}$ such that $\forall c \in C$, we have $(i', c) \in \mathcal{R}$.
iv) $\nexists c' \in \mathcal{C}$ such that $\forall i \in I$, we have $(i, c') \in \mathcal{R}$.

3.4 Generating Multiple Consensuses

This process involves filtering the FCPs based on the number of clusterings in the itemset, and build a consensus from each group. The idea is that, since the base partitions represent clustering decisions and we do not know which of these decisions is/are better than the others, then we search for the item membership similarities between different combinations of these decisions to build a final consensus decision for each possible valid combination. A *Decision Threshold* (**DT**) is used for the filtering process, as it defines the number of base clusters involved in the clustering pattern. Thus, to generate multiple consensuses, we start by building the first consensus from the instance sets of FCPs whose cluster identifier set (itemset) defines patterns shared by all base clusterings (DT= number of base clusterings[4]). Then, we sequentially decrement DT towards 1, and at each DT value, we generate a consensus from the instance sets of FCPs built from DT clusterings, plus the clusters of the previous consensus (the clusters of the consensus at DT+1).

Definition 4. *Given the first consensus $L^{MaxDT} = \{P_1, P_2, ..., P_m\}$, and the definition $B^{DT} = I^{DT} \cup L^{DT+1}$, where I^{DT} is the instance sets of the FCPs built from DT base clusterings, and L^{DT+1} is the instance sets (clusters) of the previous consensus, a new consensus L^{DT} is the result of applying a consensus function $\mathcal{Y}$ on B^{DT}, that is, $L^{DT} = \mathcal{Y}(B^{DT}) = \{L_1, L_2, ..., L_k\}$ such that $L_i \cap L_j = \emptyset$, $\forall (i, j) \in \{1, ..., k\}$, $i \neq j$, and $\bigcup_{i=1}^{i=k} L_i = \mathcal{I}$.*

[3] We can see that the number of generated clustering patterns is larger than dataset size. This happens only for a small dataset, while for a large dataset, most of its instances will share the same clustering pattern, resulting in a much lower number of patterns compared to dataset size.

[4] The clusters in the first consensus are known in [25] as "data fragments".

Table 2. Frequent Closed Patterns extracted from Table 1.

FCP ID	Itemset (FCIs)	Instance ID set
1	$\{P_1^1, P_1^2, P_1^3, P_2^4, P_1^5\}$	{3}
2	$\{P_1^1, P_2^2, P_1^3, P_2^4, P_1^5\}$	{4}
3	$\{P_2^1, P_2^2, P_1^3, P_2^4, P_1^5\}$	{5}
4	$\{P_1^1, P_1^3, P_2^4, P_1^5\}$	{3,4}
5	$\{P_2^2, P_1^3, P_2^4, P_1^5\}$	{4, 5}
6	$\{P_1^1, P_1^2, P_1^3, P_2^4, P_2^5\}$	{1,2}
7	$\{P_2^1, P_2^2, P_2^3, P_1^4, P_1^5\}$	{6,7}
8	$\{P_2^1, P_2^2, P_2^3, P_2^4, P_1^5\}$	{8,9}
9	$\{P_1^3, P_2^4, P_1^5\}$	{3,4,5}
10	$\{P_1^1, P_1^2, P_1^3, P_2^4\}$	{1,2,3}
11	$\{P_2^1, P_2^2, P_2^4, P_1^5\}$	{5,8,9}
12	$\{P_1^1, P_1^3, P_2^4\}$	{1,2,3,4}
13	$\{P_2^2, P_2^4, P_1^5\}$	{4,5,8,9}
14	$\{P_2^1, P_2^2, P_2^3, P_1^5\}$	{6,7,8,9}
15	$\{P_1^3, P_2^4\}$	{1,2,3,4,5}
16	$\{P_2^4, P_1^5\}$	{3,4,5,8,9}
17	$\{P_2^1, P_2^2, P_1^5\}$	{5,6,7,8,9}
18	$\{P_2^2, P_1^5\}$	{4,5,6,7,8,9}
19	$\{P_2^4\}$	{1,2,3,4,5,8,9}
20	$\{P_1^5\}$	{3,4,5,6,7,8,9}

At each DT, an instance set $I \subseteq \mathcal{I}$ has one of the following three properties:

i) Uniqueness: it does not intersect with any other set $I' \subseteq \mathcal{I}$, that is, $I \cap I' = \emptyset$.
ii) Inclusion: it is a subset of another set $I' \subseteq \mathcal{I}$, that is, $I \subseteq I'$.
iii) Intersection: it intersects with another set $I' \subseteq \mathcal{I}$, that is, $I \cap I' \neq \emptyset$, $I \setminus I' \neq \emptyset$ and $I' \setminus I \neq \emptyset$.

Note that at the first consensus, all the instance sets are unique, because they are generated from clustering patterns shared by all base clusterings. However, for the next consensuses, the sets of instance identifiers can have any of the above properties, because when we consider fewer base clusterings, instances can belong to several patterns.

The objective of the consensus function is to build unique clusters from sets of instance identifiers. Thus, to decide how to deal with intersecting sets, we use the size of intersection as a measure for merging or splitting them. The idea of measuring the similarity between sets based on their intersection is not new. Jaccard index [14] is based on calculating the ratio between intersection size over the size of the union of 2 sets:

$$Jaccard(X, Y) = \frac{|X \cap Y|}{|X \cup Y|} = \frac{|X \cap Y|}{|X| + |Y| - |X \cap Y|}$$

Let us consider the three cases of sets intersection in Fig. 1. By calculating Jaccard for each case we get:

$$Jaccard(A,B) = \frac{3}{27} \text{ , } Jaccard(B,C) = \frac{7}{23} \text{ , } Jaccard(D,E) = \frac{7}{23}$$

Jaccard measure provides the same score for cases 2 and 3, while they are actually different: Indeed, the set B is mostly part of set C while the sets in case 3 share only about the half of their instances. Thus, instead of using Jaccard, we take a merge/split decision based on comparing intersection size over the size of each set. Let us then define $I(X|Y)$ as the ratio of intersection between sets X and Y over the size of set X, that is:

$$I(X|Y) = \frac{|X \cap Y|}{|X|}$$

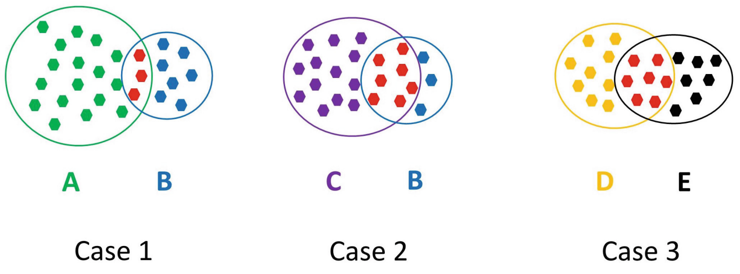

Fig. 1. Examples of sets intersection

Case 1: $I(B|A) = \frac{|A \cap B|}{|B|} = \frac{3}{10}$, $I(A|B) = \frac{|A \cap B|}{|A|} = \frac{3}{20}$

Case 2: $I(B|C) = \frac{|B \cap C|}{|B|} = \frac{7}{10}$, $I(C|B) = \frac{|B \cap C|}{|C|} = \frac{7}{20}$

Case 3: $I(E|D) = \frac{|E \cap D|}{|E|} = \frac{7}{14}$, $I(D|E) = \frac{|D \cap E|}{|D|} = \frac{7}{16}$

From the above scores, we can see that $I(B|C)$ is the highest, and that the scores provided for cases 2 and 3 are different, compared to Jaccard which assigned them the same score. Thus the question is: how will we use all these information to decide to merge or split intersecting sets? We propose the following method:

To decide to either merge or split intersecting sets, a *Merging Threshold* (**MT**) can be used. MT is the minimum intersection ratio of a set (X) to decide to merge it with another set (Y). That is, sets X and Y are merged only if the intersection ratio of any of them, $I(X|Y)$ or $I(Y|X)$, is bigger than MT, otherwise they are split. While the merge operation is simply the union of two sets, the split operation involves removing the common instances from one of the sets. Since the sets represent clusters, the winner is the smaller set, as it is fundamentally more coherent. Thus, the shared instances are kept in the smaller cluster and removed from the bigger one. Any set that is subset of another set is removed. The process checks all the sets and the newly generated sets of the merge/split operation until obtaining all the remaining sets as unique sets. Algorithm 1 explains the full process of generating multiple consensuses.

Going back to the FCPs in Table 2, the first consensus (DT=5) consist of 6 clusters (data fragments) which are the instance sets of FCPs 1, 2, 3, 6, 7, and 8. For the next consensus (DT=4), we add the instance sets of FCPs 4, 5, 10, 11, and 14. However, the clusters of the previous consensus will be removed because they are just subsets of the new sets. Therefor, we will have the following sets: {3,4}, {4,5}, {1,2,3}, {5,8,9}, and {6,7,8,9}. With MT=0.5, we start with set {3,4} that intersects with set {4,5} by 0.5 of their instances, thus they are merged to form the set {3,4,5}. Next, set {3,4,5} intersects with set {1,2,3} but by 0.3 which is less than MT, thus they are split into sets {3,4,5} and {1,2}. The same split process is performed for sets {3,4,5} and {5,8,9} to form sets {3,4,5} and {8,9}. For the remaining sets, we find that set {8,9} is a subset of {6,7,8,9}, thus it is removed. The final clusters at DT=4 are: {3,4,5}, {1,2}, and {6,7,8,9}. The same process is performed for the remaining DTs.

By varying DT, it is possible that a consensus at a given value is identical to the previous one. Therefore, redundant consensuses are removed, and a *Stability counter* (**ST**) is used to record how many times a consensus is generated. The ST value is assigned to the consensus with the highest DT, suggesting that there is no better solution found for ST consecutive consensuses. Although the recommended consensus is the one with the highest similarity to the ensemble, a stable consensus can also be considered as another good solution.

3.5 ConsTree

The ConsTree is a Hasse diagram of consensuses, where each level depicts the clusters of a consensus as nodes, with node's size and label reflecting the cluster size. The bottom of the tree is the first consensus, then the tree goes up to the root having, at maximum, number of levels equals the number of base clusterings. Each cluster in a consensus can be linked to several clusters at the higher level, because the merge/split operations can result in regrouping some instances in a different manner at a higher level. Figure 2 shows a ConsTree of applying 9 base clusterings to partition a dataset of 400 instances, with the recommended consensus circled by a red line.

Definition 5. *A tree of consensuses is an ordered set ($\mathcal{L}$, $\preceq$) of consensuses $\mathcal{L} = \bigcup_{DT=MaxDT}^{DT=MinDT} L^{DT}$ ordered in descending order of DT values. Let us denote $L^{\alpha} = \{P_1^{\alpha}, ..., P_m^{\alpha}\}$ and $L^{\beta} = \{P_1^{\beta}, ..., P_n^{\beta}\}$ the consensuses generated for α and β DT values respectively. Let us denote P_q^{α} the q^{th} cluster in L^{α} and P_r^{β} the r^{th} cluster in L^{β}, with $1 \leq q \leq m$ and $1 \leq r \leq n$. For $\alpha > \beta$ we have $L^{\alpha} \preceq L^{\beta}$, that is $\forall P_q^{\alpha} \in L^{\alpha}$, $\exists P_r^{\beta} \in L^{\beta}$ such that $P_q^{\alpha} \cap P_r^{\beta} \neq \emptyset$. L^{α} is a predecessor of L^{β} in the tree of consensuses.*

In the tree of Fig. 2, few shifting of instances occur, while the majority is just merging clusters from low level into 1 cluster at a higher level. However,

```
Input   : Dataset to cluster, merging threshold $MT$
Output  : ConsTree tree of consensuses, list of consensus clustering vectors
Generate clusterings ensemble of the dataset;
Build the cluster membership matrix $\mathcal{M}$;
Generate FCPs from $\mathcal{M}$ for $minsupport = 0$;
Sort the FCPs in ascending order of the size of their instance set;
$MaxDT \leftarrow$ Number of base clusterings;
$BiClust \leftarrow$ {instance sets of FCPs built from $MaxDT$ base clusters};
Assign a label to each set in $BiClust$ to build the first consensus vector and store it in a
list of vectors $ConsVctrs$;
for $DT = (MaxDT - 1)$ to $1$ do
    $BiClust \leftarrow BiClust \cup$ {instances sets of FCPs built from $DT$ base clusters};
    $N \leftarrow |BiClust|$ ;
    repeat
        for $i = 1$ to $N$ do
            $B_i \leftarrow i^{\text{th}}$ set in $BiClust$;
            for $j = 1$ to $N$, $j \neq i$ do
                $B_j \leftarrow j^{\text{th}}$ set in $BiClust$;
                $IntrscLeng \leftarrow |B_i \cap B_j|$;
                if $IntrscLeng = 0$ then
                    Next j ;
                else if $IntrscLeng = |B_i|$ then
                    Remove $B_i$ from $BiClust$;
                    Next $i$;
                else if $IntrscLeng = |B_j|$ then
                    Remove $B_j$ from $BiClust$;
                    Next $j$;
                else if $(IntrscLeng \geq |B_i| \times MT)$ or $(IntrscLeng \geq |B_j| \times MT)$
                then
                    $B_j \leftarrow B_i \cup B_j$;
                    Remove $B_i$ from $BiClust$;
                    Next $i$;
                else
                    if $|B_i| \leq |B_j|$ then
                        $B_j \leftarrow B_j \setminus B_i$ ;
                    else
                        $B_i \leftarrow B_i \setminus B_j$ ;


            end
        end
    until All sets in BiClust are unique;
    Assign a label to each set in $BiClust$ to build a consensus vector and add it to
    $ConsVctrs$;
end
Find stable consensuses in $ConsVctrs$ and remove extra duplicates;
For each remaining consensus, calculate its average similarity to the ensemble using
Jaccard index;
Build a tree from the consensuses in $ConsVctrs$, with a recommended solution as the
one that has the highest average similarity to the ensemble;
```

Algorithm 1. Generate Multiple Consensuses

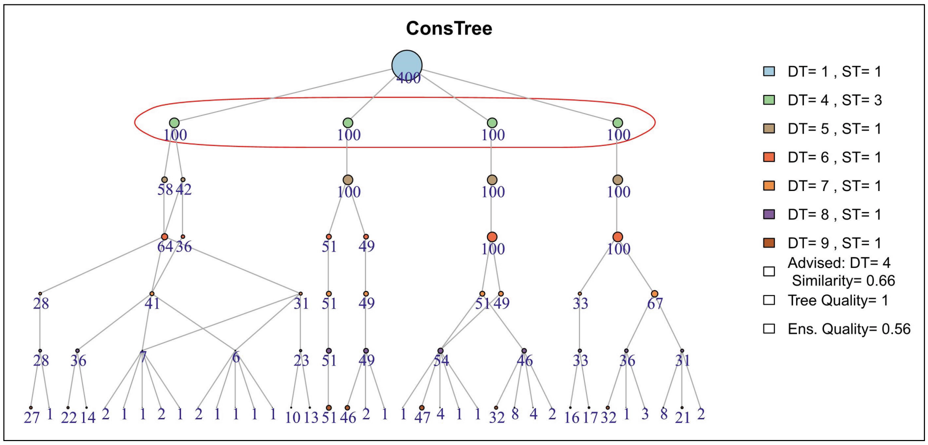

Fig. 2. Example of a ConsTree.

to enhance the visualization of the tree and to make it easier for the analyst to recognize the stable clusters or their cores (those that do not change over several tree levels), we developed a *tree refinement process* on which the few instances that shift are removed. Figure 3 is the result of performing the refinement process on the tree in Fig. 2. The refinement process does not alter the original consensuses, it just simplify the visualization. The *Removed Instances* (RI) at the bottom of the figure tells how many instances are removed. It is a set of instance identifiers that the analyst can retrieve to investigate why these instances are not stable and cause conflicts between the base clusterings on where they should belong. If she/he prefers the clusters generated by the refinement process, she/he can simply remove the RI from the selected consensus.

The tree represent an important tool to analyze the dataset and discover the hidden cluster structure. For example, we can recognize in Fig. 2 4 columns of node-structures (the heads of these columns are the children nodes of the root). The ST value of DT=4 (which happened to be also the recommended solution) tells that this consensus of 4 clusters is the most stable one, which also adds to our discovery of a hidden structure of 4 clusters. We can also recognize other strong clusters, as the clusters of sizes 51 and 49 in the second column from the left, telling a strong intra-cluster similarity between their instances compared to other clusters. The fact that these 2 clusters are then merged into 1 cluster tells that their instances are close in the data space. The refinement process allowed us also to discover in Fig. 3 that 36 instances have strong similarity between them so they did not change over 4 consecutive tree levels. Based on such analysis, the analyst is not restricted to choose the recommended consensus, since the meaningfulness of the clusters depends more on the analyst preference. This is why we generate multiple consensuses from different combinations of base clusters (different views), rather than presenting just one solution.

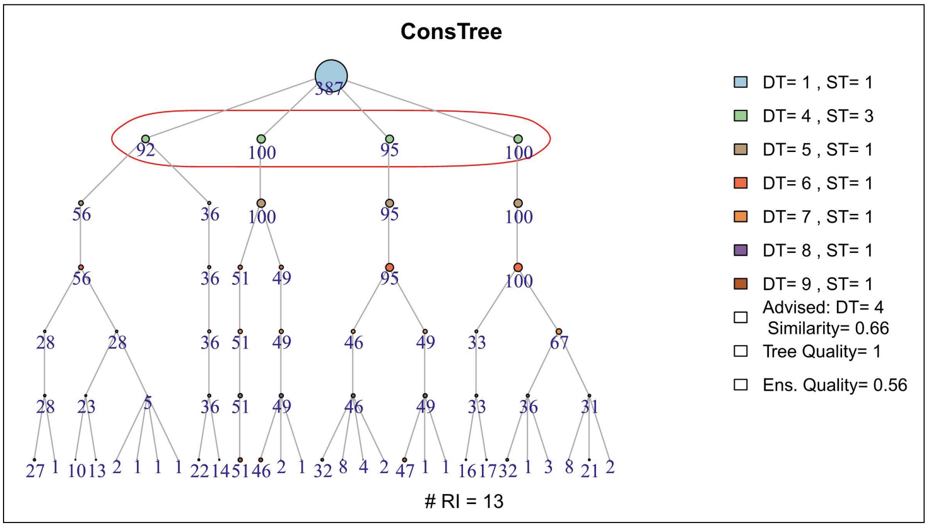

Fig. 3. The result of refining the tree in Fig. 2.

4 Experiments

Tests were run on a laptop with Intel® Core$^{\text{TM}}$ i7-4710MQ @ 2.50GHz, and 32 GB RAM. The proposed method was implemented using R language [18]. To find the clustering patterns, function *apriori* in *arules* R package [11] was used, by setting the *target* parameter to "closed frequent itemsets".[5] To draw the ConsTree, each cluster in a consensus was represented as a node in a graph. Nodes of consecutive consensuses were linked by edges based on the shared instances between them. A data frame that defines these edges was used to build the graph, then the ConsTree was plotted by the *plot* function in the *igraph* R package [3]. The refinement process keeps only the edges that has the maximum number of shared instances between 2 nodes.

Table 3 summarizes the performed tests. In each test, the ensemble was generated by random selection of the following clustering algorithms with random settings: K-means, PAM, agglomerative hierarchical clustering, AGNES, DIANA, MCLUST (Gaussian Model-Based Clustering), C-Means, FANNY, Bagged Clustering, and SOM. All these clustering algorithms are available in R. If the random generation of the ensemble results in creating identical clusterings, 1 cluster , or 1 dominant cluster that involves 90% of the instances, then these partitions are removed. "Ensemble size" specifies the number of base clusterings, that each partitioned the dataset into a number of clusters within the range "K range". We measured the quality of the ensemble as the average Jaccard similarity between

[5] A faster algorithm for generating closed itemsets called FIST is proposed by [16], and an implementation of it in java is available on the website of the authors.

Table 3. Tests validation.

Dataset	Seeds	Mushroom	Zoo	E.Coli	Iris	Breast Cancer	Wine	EngyTime	Wingnut
Dataset Size	210	8124	101	336	150	699	178	4096	1016
# of attributes	7	22	16	7	4	9	13	2	2
# of true classes	3	2	7	8	3	2	3	2	2
Ensemble size	9	7	10	12	8	9	9	8	12
K range	[2,8]	[2,5]	[2,12]	[4,10]	[2,8]	[2,6]	[2,8]	[2,8]	[2,7]
In-ensemble similariy	0.56	0.58	0.62	0.59	0.53	0.67	0.58	0.49	0.58
Ensemble Min.	0.30	0.34	0.45	0.28	0.29	0.42	0.45	0.27	0.37
Ensemble Max.	0.74	0.69	0.90	0.69	0.73	0.82	0.89	0.88	1.00
Our method	0.74	0.67	0.84	**0.67**	0.69	**0.87**	0.78	**0.86**	**0.98**
# of clusters in our method	3	3	5	5	3	2	3	2	2
SE	0.64	**0.70**	0.75	0.42	0.62	**0.87**	0.82	**0.86**	0.88
GV1	0.67	**0.70**	0.82	0.38	**0.71**	0.86	0.80	0.85	0.88
DWH	0.67	**0.70**	0.81	0.51	0.62	0.85	0.73	0.75	0.89
HE	0.65	**0.70**	0.81	0.50	0.62	0.85	0.78	**0.86**	0.88
GV3	0.72	**0.70**	0.82	0.44	0.59	**0.87**	**0.84**	**0.86**	0.91
SM	0.64	**0.70**	0.79	0.50	0.62	0.85	0.78	**0.86**	0.88
soft/symdiff	**0.75**	**0.70**	0.69	0.42	0.59	0.86	0.47	**0.86**	0.92
Medoids	0.74	0.69	**0.90**	0.64	0.55	0.81	0.75	0.64	0.52

each pair of clusterings in the ensemble. We call this the "in-ensemble similarity", while "ensemble min" and "ensemble max" denotes the minimum and maximum similarity to the true class. These information are to ensure that we did not use very similar high quality clusterings in the ensemble to generate a high quality consensus. In all tests, we compared the "recommended consensus" of our method against voting-based consensus methods available in R package CLUE [13], which include the following: SE, GV1, DWH, HE, SM, GV3, soft/symdiff, and consensus medoid. To justify the quality of the results, the consensus solution is compared using Jaccard index against true class labels available for each tested dataset. Note that CLUE methods require specifying the number of required clusters in the consensus, thus we used the true number of classes, while our method do not require this. For the MT parameter, we used MT = 0.5 as the default value.

Seeds, Mushroom, Zoo, E.Coli, Iris, Breast Cancer and Wine are real datasets available on the UCI repository [15]. EngyTime and Wingnut are synthetic datasets from [23]. Table 4 shows the execution time, in seconds, of the consensus methods used. For our method, we separated between the time required to generate the patterns, and the time used by the method to generate all the consensuses and calculate the recommended one.[6] We can see that the total time of the proposed method is acceptable, and it does not depend on the dataset

[6] The time required to display the ConsTree is not considered, as it depend on the I/O device.

Table 4. Execution time of the consensus methods (in seconds).

Dataset	Seeds	Mushroom	Zoo	E.Coli	Iris	Breast Cancer	Wine	EngyTime	Wingnut
Patterns	0.116	0.353	0.085	0.342	0.053	0.147	0.147	0.210	0.568
Our method	0.417	0.491	0.262	1.492	0.229	0.488	0.688	3.567	1.892
SE	0.011	0.064	0.010	0.037	0.013	0.017	0.009	0.066	0.023
GV1	0.080	0.047	0.048	0.076	0.019	0.018	0.026	0.051	0.031
DWH	0.006	0.051	0.006	0.010	0.016	0.009	0.006	0.030	0.015
HE	0.011	0.062	0.017	0.018	0.001	0.016	0.008	0.087	0.020
GV3	1.483	3532.199	0.664	7.061	0.768	13.031	0.848	673.044	30.901
SM	0.722	22.917	0.817	9.305	0.675	2.172	0.833	11.455	4.720
soft/symdiff	10.918	21925.92	4.433	55.181	5.301	175.815	8.518	5332.099	414.718
Medoids	0.028	0.222	0.031	0.067	0.016	0.047	0.030	0.154	0.101

size, but on the number of the base clusterings used, and the similarity between them. For example, in the test of the Mushroom dataset, the total number of generated patterns is 106, while the dataset size is 8124 instances. This is a huge pruning of the search space.

5 Conclusions

We presented a new method that can generate multiple consensus clustering solutions, and recommend to the user the solution the most similar to the ensemble. Frequent closed itemsets technique is used to detect similarities between the base partitions, and define clustering patterns common to sets of instances. The similarity between clustering patterns is calculated based on their shared instances. The tests showed that the proposed method achieved generally good results in terms of quality and the number of discovered clusters. In addition, it does not require the user to specify K (the number of clusters in the generated consensus), as this parameter is difficult to predict in the absence of domain knowledge about the number of hidden clusters in the dataset.

An additional advantage of the proposed method is the ConsTree. As an analysis tool, it enables the user to discover strong clusters, that is, those that do not change over several tree levels, pointing out to strong intra-cluster similarity among the instances. A stable consensus (identified on the tree by ST>1) suggests usually the existence of a well separated clusters structure, thus the user is advised to investigate this consensus in addition to the recommended solution.

Execution time of our method is not related directly to the dataset size as in other consensus methods. Instead, it depends on the size of the ensemble and whether there are many similarities or conflicts among the base clusterings, as this will determine the generated clustering patterns. Thus, for large datasets, we can get smaller number of patterns compared to dataset size if there are many agreements between the base clusterings. In addition, to generate a consensus, the proposed method requires only accessing a small subset of the available clustering patterns. Tests showed that the CLUE methods "GV3" and "soft/symdiff" cost longer execution time compared to all other methods, and are not applicable on large datasets because of their high storage complexity.

References

1. Asur, S., Ucar, D., Parthasarathy, S.: An ensemble framework for clustering protein-protein interaction networks. Bioinformatics **23**(13), i29–i40 (2007)
2. Ceglar, A., Roddick, J.F.: Association mining. ACM Computing Surveys **38**(2) (2006)
3. Csardi, G., Nepusz, T.: The igraph software package for complex network research. InterJournal Complex Systems **1695** (2006). http://igraph.org
4. Dalton, L., Ballarin, V., Brun, M.: Clustering algorithms: on learning, validation, performance, and applications to genomics. Current Genomics **10**(6), 430 (2009)
5. Dudoit, S., Fridlyand, J.: Bagging to improve the accuracy of a clustering procedure. Bioinformatics **19**(9), 1090–1099 (2003)
6. Färber, I., Günnemann, S., Kriegel, H.P., Kröger, P., Müller, E., Schubert, E., Seidl, T., Zimek, A.: On using class-labels in evaluation of clusterings. In: KDD MultiClust International Workshop on Discovering, Summarizing and Using Multiple Clusterings, p. 1 (2010)
7. Fern, X.Z., Brodley, C.E.: Solving cluster ensemble problems by bipartite graph partitioning. In: Proceedings of the Twenty-First International Conference on Machine Learning, p. 36. ACM (2004)
8. Fischer, B., Buhmann, J.M.: Bagging for path-based clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence **25**(11), 1411–1415 (2003)
9. Fred, A.L., Jain, A.K.: Combining multiple clusterings using evidence accumulation. IEEE Transactions on Pattern Analysis and Machine Intelligence **27**(6), 835–850 (2005)
10. Ghaemi, R., Sulaiman, M.N., Ibrahim, H., Mustapha, N.: A survey: Clustering ensembles techniques. WASET **50**, 636–645 (2009)
11. Hahsler, M., Gruen, B., Hornik, K.: arules – A computational environment for mining association rules and frequent item sets. Journal of Statistical Software **14**(15), 1–25 (2005)
12. Halkidi, M., Batistakis, Y., Vazirgiannis, M.: On clustering validation techniques. Journal of Intelligent Information Systems **17**(2), 107–145 (2001)
13. Hornik, K.: A CLUE for CLUster Ensembles. Journal of Statistical Software **14**(12) (2005)
14. Jaccard, P.: The distribution of the flora in the alpine zone.1. New Phytologist **11**(2), 37–50 (1912). http://dx.doi.org/10.1111/j.1469-8137.1912.tb05611.x
15. Lichman, M.: UCI machine learning repository (2013). http://archive.ics.uci.edu/ml
16. Mondal, K.C., Pasquier, N., Mukhopadhyay, A., Maulik, U., Bandhopadyay, S.: A new approach for association rule mining and bi-clustering using formal concept analysis. In: Perner, P. (ed.) MLDM 2012. LNCS, vol. 7376, pp. 86–101. Springer, Heidelberg (2012)
17. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Efficient mining of association rules using closed itemset lattices. Inf. Systems **24**(1), 25–46 (1999)
18. R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2015). https://www.R-project.org/

19. Rendón, E., Abundez, I., Arizmendi, A., Quiroz, E.: Internal versus external cluster validation indexes. International Journal of Computers and Communications **5**(1), 27–34 (2011)
20. Sarumathi, S., Shanthi, N., Sharmila, M.: A comparative analysis of different categorical data clustering ensemble methods in data mining. IJCA **81**(4), 46–55 (2013)
21. Strehl, A., Ghosh, J.: Cluster ensembles – a knowledge reuse framework for combining multiple partitions. JMLR **3**, 583–617 (2003)
22. Topchy, A., Jain, A.K., Punch, W.: Clustering ensembles: Models of consensus and weak partitions. IEEE Transactions on Pattern Analysis and Machine Intelligence **27**(12), 1866–1881 (2005)
23. Ultsch, A.: Clustering with SOM: U*C. In: Proc. WSOM Workshop, pp. 75–82 (2005)
24. Vega-Pons, S., Ruiz-Shulcloper, J.: A survey of clustering ensemble algorithms. IJPRAI **25**(03), 337–372 (2011)
25. Wu, O., Hu, W., Maybank, S.J., Zhu, M., Li, B.: Efficient clustering aggregation based on data fragments. IEEE Trans. Syst. Man Cybern. B Cybern. **42**(3), 913–926 (2012)