Frequent Closed Patterns Based Multiple Consensus Clustering

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. Clustering is one of the major tasks in data mining. However, selecting an algorithm to cluster a dataset is a difficult task, especially if there is no prior knowledge on the structure of the data. Consensus clustering methods can be used to combine multiple base clusterings into a new solution that provides better partitioning. In this work, we present a new consensus clustering method based on detecting clustering patterns by mining frequent closed itemset. Instead of generating one consensus, this method both 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 patterns

1 Introduction

Clustering is the process of partitioning a dataset into groups, so that the instances in the same group are more similar to each other than to instances in any other group. This partitioning may lead to discover meaningful patterns in the dataset. Many clustering algorithms were developed in the last 50 years, and, most often, each algorithm produces different partitioning when applied to the same dataset, because they are designed to target a specific model (compact clusters, non-convex clusters...). Thus the question is: How to choose a clustering for a dataset from these many possibilities?

The most common solution is to use validation measure(s) to compare the results and select the one that gets the higher score [4,7]. There are two general categories of validation measures: *Internal validation* that compares the clustering against a specific clustering model, and *external validation* that compares the clustering against true labels (class labels given on an evaluation set using domain knowledge). In both categories, many validation measures exist, and no one impartially evaluates the results of all clustering algorithms [20]. In real life, the user can have similar scores for different validation measures and/or for different clustering results, while the results are different in many aspects, like in the number of clusters or in the instance grouping into clusters.

L. Rutkowski et al. (Eds.): ICAISC 2016, Part II, LNAI 9693, pp. 14–26, 2016.
DOI: 10.1007/978-3-319-39384-1_2

Rather than depending on validation measures, another approach is to combine the multiple clustering solutions generated by several clustering algorithms and/or settings, in order to produce a final clustering which is better than each individual algorithm can produce. This technique is called *consensus clustering*, *aggregation of clusterings* or *ensemble clustering*, and the clustering algorithms to be combined are called *base clustering algorithms*. Many consensus clustering methods have been proposed, and some of them will be discussed in the next section. In this paper, we propose a new consensus clustering method named MultiCons. Instead of providing the user with a single solution, we generate multiple consensuses by varying the selection of base clusterings, then linking these multiple solutions in a hierarchical view. The user can then not only select the best solution, but also discover strong clusters in the dataset that do not change when varying the base clusterings. Hence, our proposed method is a combination of ensemble of consensus solutions with a visual data analysis tool.

The paper is organized as follows: Sect. 2 discusses some of the previous work in consensus clustering. Section 3 explains the proposed approach. Some experimental results are shown in Sect. 4, and we provide conclusions in Sect. 5.

2 Related Work

Consensus clustering refers to the problem of finding a single consensus clustering from a number of different inputs or base clusterings that have been obtained for a given dataset [11,24]. The advantage of this technique is to have a new clustering result that is at least as good as the best clustering achieved by the base methods.

Many consensus clustering methods were developed over the past years. In Asur *et al.* [1], six predefined clustering algorithms suitable for protein-protein datasets clustering were considered as base clusterings. A cluster membership matrix[1] is then built, and a consensus clustering method is applied over this matrix (agglomerative hierarchical clustering or recursive bisection) to obtain the final consensus. All the 6 base clusterings have K clusters, and if K is high the resulting binary membership matrix is sparse and the consensus clustering of this matrix is ineffective. Thus, PCA (Principle Components Analysis) is applied before the consensus clustering to reduce the dimensionality of the membership matrix into less, but more expressive, dimensions. These different consensus techniques were compared, and the authors conclude that the PCA based technique produced very efficient clustering and identified multiple functionalities of proteins.

Three consensus clustering methods were proposed by Strehl & Ghosh [18]. The first step for all their proposed consensus functions is to transform the given clusterings into a suitable hypergraph representation, where each cluster (column in the membership matrix) from any base clustering is considered as a hyperedge that connects several vertices (all the instances in this cluster) in a hypergraph. Based on this mapping, Strehl & Ghosh propose: (i) *Cluster-based Similarity*

[1] See Sect. 3.1 for a definition of cluster membership matrix.

Partitioning Algorithm (CSPA) which is based on an overall similarity matrix S built from the membership matrix H by using $S = \frac{1}{r}HH^{\dagger}$, where r is the number of base clusterings. The aforementioned hypergraph is built from this similarity matrix so that each hyperdge represents the sum of similarities between a given pair of vertices (i.e. each time the two considered vertices are clustered together by any base clustering, their similarity is increment by 1), then a graph-based clustering method (METIS) provides the consensus clustering; (ii) *HyperGraph-Partitioning Algorithm* (HGPA) all hyperedges as well as all vertices are equally weighted, then a hypergraph partitioning algorithm (HMETIS) defines the consensus by cutting a minimal number of edges; (iii) *Meta-CLustering Algorithm* (MCLA) follow the same ideas, but hyperedges weights are proportional to the similarity between vertices (instances) which is calculated using binary Jaccard measure. The resulting hypergraph is then clustered using METIS to generate K clusters. For each of these K meta-clusters, its hyperedges are collapsed into a single meta-hyperedge. Each meta-hyperedge has an association vector which contains an entry for each object describing its level of association with the corresponding meta-cluster.

With their algorithm *WClustering*, Li & Ding [11] proposed weighting the base clusterings to ensure removing redundant (similar) partitions, since this process produces better results compared to other methods that generate the consensus from brute-force averaging of the base clusterings. Weights are automatically determined by an optimization process. Experimental results showed that more accurate clustering was achieved by the k-means algorithm when applied to the weighted consensus similarity matrix, compared to the results of CSPA and HGPA. In Zhang & Li [24], the base clusterings are compared using pairwise similarity, and then divided into groups using K-means. On each group, one of the previously discussed consensus methods is used: PCA-based consensus algorithm [1], CSPA and HGPA from [18], and WClustering [11]. Thus, the final result is K consensuses for the user to select from.

The idea in Caruana *et al.* [2] is to generate many base clusterings, then build a similarity matrix for these different partitions using Rand index. This similarity matrix is passed to agglomerative hierarchical clustering to build a meta clustering. The dendrogram shows how the partitions are similar to each other, thus there is no final consensus. Instead, the user can analyze the resulting dendrogram to choose which clustering is the most relevant. To have a diversity in the base clusterings, feature weighting using Zipf distribution and PCA were used to produce different base clustering views.

For more information about consensus clustering methods, see Ghaemi *et al.* [5], Sarumathi *et al.* [17], and Vega-Pons & Ruiz-Shulcloper [20].

3 The Proposed Approach

Unlike consensus clustering methods that search for a *median partition* [20] to enhance the results of an ensemble of base clustering results, our objective is to identify hidden cluster structure in the dataset from the ensemble. Using the

Frequent Closed Itemsets (FCIs) [15] technique from pattern mining domain, we discover clustering patterns common to different sets of base clusterings. FCIs define both clustering patterns (fragments) common to all base clusterings, known as *data fragments* in [21], plus other "larger fragments" built from fewer base clusterings. Regrouping these fragments based on the number of base clusterings used to define them, patterns in each group can then be combined according to their common instances to build a consensus. Generated consensus clusters are then linked in a tree-shaped diagram to easily understand their building process and identify stable clusters. Algorithm 1 describes the successive steps of the proposed approach, as explained in the following subsections.

3.1 Cluster Membership Matrix

From the multiple clusterings of the dataset generated using a set of base clustering methods, a *cluster membership matrix* $\mathcal{M}$ is built. $\mathcal{M}$ is a binary matrix of $N \times M$ cells, where N is the number of dataset instances, and M is the number of cluster vectors (total number of clusters generated by all base clustering algorithms), 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 variables, each designating a cluster, 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 the instance i belongs to the cluster c.*

Consider for example a dataset of nine instances $\mathcal{D} = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ partitioned using five base clusterings into the five following partitions: $P1 = \{\{1, 2, 3\}, \{4, 5, 6, 7, 8, 9\}\}$, $P2 = \{\{1, 2, 3\}, \{4, 5, 6, 7, 8, 9\}\}$, $P3 = \{\{1, 2, 3, 4, 5\}, \{6, 7\}, \{8, 9\}\}$, $P4 = \{\{4, 5, 6, 7\}, \{1, 2, 3\}, \{8, 9\}\}$, and $P5 = \{\{4, 5, 6, 7\}, \{1, 2, 3\}, \{8, 9\}\}$. Table 1 shows the resulting cluster membership matrix consisting of 9 rows (instances) and 14 columns (total number of clusters in base clusterings). 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 hereafter.

Definition 2. *An item of a cluster membership matrix $\mathcal{M}$ = ($\mathcal{I}$, $\mathcal{C}$, $\mathcal{R}$) is a cluster identifier $c \in \mathcal{C}$ and 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 $\mathcal{M}$ iff 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.*

3.2 Generating Clustering Patterns

The next step consists of generating the *Frequent Closed Patterns* (FCPs) from the cluster membership matrix $\mathcal{M}$. Each FCP associates a FCI (a closed set of cluster identifiers) and its corresponding set of instance identifiers, i.e., the

Table 1. Example cluster membership matrix.

Instance ID	P_1^1	P_2^1	P_1^2	P_2^2	P_1^3	P_2^3	P_3^3	P_1^4	P_2^4	P_3^4	P_1^5	P_2^5	P_3^5
1	1	0	1	0	1	0	0	0	1	0	0	1	0
2	1	0	1	0	1	0	0	0	1	0	0	1	0
3	1	0	1	0	1	0	0	0	1	0	0	1	0
4	0	1	0	1	1	0	0	1	0	0	1	0	0
5	0	1	0	1	1	0	0	1	0	0	1	0	0
6	0	1	0	1	0	1	0	1	0	0	1	0	0
7	0	1	0	1	0	1	0	1	0	0	1	0	0
8	0	1	0	1	0	0	1	0	0	1	0	0	1
9	0	1	0	1	0	0	1	0	0	1	0	0	1

identifiers of dataset instances that are common to all clusters in the FCI set. FCPs represent maximal sets, regarding inclusion, of base clusterings that agree on grouping a set of instances. Subsets of such sets of base clusterings that agree on grouping the same set of instances will not be considered.[2] Stated another way, FCPs are maximal rectangles in the membership matrix. See [23] for complexity considerations about FCIs and frequent pattern mining.

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* $(i, c') \in \mathcal{R}$.

Table 2 shows the set of the seven FCPs extracted from Table 1. Each row, identified by its FCP ID, represents an FCP. The support of each FCP corresponding to the size of its instance set, the *minsupport* parameter is set to 0 in order to consider clustering patterns, i.e., clusters of base clusterings, of all sizes.

3.3 Generating Multiple Consensuses

Building clustering consensuses from the FCPs is an iterative process that considers during each iteration all FCPs corresponding to a specific number, called *Decision Threshold* (DT), of base clusterings. The DT value represents the minimum number of base clusterings to consider for building a consensus. For the first consensus, we have DT = *MaxDT*, where *MaxDT* is the number of base clusterings used. The DT value is then sequentially decremented until DT = 1 to integrate in the new consensus another clustering view generated by a smaller

[2] Generating only clustering patterns of maximum agreement between base clusterings reduces processing time.

Table 2. Frequent closed patterns extracted from Table 1.

FCP IDs	Itemsets (FCIs)	Instance IDs
1	$\{P_2^1, P_2^2, P_1^3, P_1^4, P_1^5\}$	{4, 5}
2	$\{P_2^1, P_2^2, P_2^3, P_1^4, P_1^5\}$	{6, 7}
3	$\{P_2^1, P_2^2, P_3^3, P_3^4, P_3^5\}$	{8, 9}
4	$\{P_1^1, P_1^2, P_1^3, P_2^4, P_2^5\}$	{1, 2, 3}
5	$\{P_2^1, P_2^2, P_1^4, P_1^5\}$	{4, 5, 6, 7}
6	$\{P_1^3\}$	{1, 2, 3, 4, 5}
7	$\{P_2^1, P_2^2\}$	{4, 5, 6, 7, 8, 9}

number of base clusterings. During an iteration DT $= n$, all FCPs with an FCI of size n are combined with clusters of the previous iteration (for DT $= n + 1$) making each consensus a complete clustering vector, i.e., covering all dataset instances:

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

The first consensus consists of instance sets of FCPs that define clustering patterns common to all base clusterings (lines 5–7 in Algorithm 1), or data fragments [21]. Consensuses are then iteratively built using results of the previous consensus (lines 9–32 in Algorithm 1) according to the following properties of instance sets. 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 $I' \setminus I \neq \emptyset$.

To build a new consensus, intersecting sets, that represent sets of instances that are close (very similar) in the data space,[3] of FCPs and clusters are merged (lines 23–27 in Algorithm 1). Instance sets having *inclusion* property are removed to consider the new clustering view (lines 17–22 in Algorithm 1). The merging process repeats until all instance sets have *uniqueness* property. Since only a small number of FCPs is used for generating each consensus, the process of generating consensuses is efficient, even when the total number of FCPs is large.

[3] This is the objective of clustering algorithms, yet they differ in how they define the similarity between instances.

```
Input   : Dataset to cluster.
Output: ConsTree tree of consensuses.
1 Generate multiple base clusterings of the dataset;
2 Build the cluster membership matrix $\mathcal{M}$;
3 Generate FCPs from $\mathcal{M}$ for $minsupport = 0$;
4 Sort the FCPs in ascending order of the size of their instance list;
5 $MaxDT \leftarrow$ Number of base clusterings;
6 $BiClust \leftarrow$ {instance sets of FCPs built from $MaxDT$ base clusters};
7 Assign a label to each set in $BiClust$ to build the first consensus vector and
  store it in a list of vectors $ConsVctrs$;
8 /* Build the remaining consensuses */;
9 for $DT$ = ($MaxDT$ - 1) to 1 do
10     $BiClust \leftarrow BiClust \cup$ {instance sets of FCPs built from $DT$ base clusters};
11     $N \leftarrow |BiClust|$   // Nbr of sets in $BiClust$;
12     repeat
13         for $i$ = 1 to $N$ do
14             $B_i \leftarrow i^{\text{th}}$ set in $BiClust$;
15             for $j$ = 1 to $N$, $j \neq i$ do
16                 $B_j \leftarrow j^{\text{th}}$ set in $BiClust$;
17                 if $B_i \subseteq B_j$ then
18                     Remove $B_i$ from $BiClust$;
19                     Next $i$;
20                 else if $B_j \subset B_i$ then
21                     Remove $B_j$ from $BiClust$;
22                     Next $j$;
23                 else if $B_i \cap B_j \neq \emptyset$ then
24                     $B_j \leftarrow B_i \cup B_j$;
25                     Remove $B_i$ from $BiClust$;
26                     Next $i$;
27                 end
28             end
29         end
30     until All sets in BiClust are unique;
31     Assign a label to each set in $BiClust$ to build a consensus vector and add it
       to $ConsVctrs$;
32 end
33 /* Remove similar consensuses */;
34 $ST \leftarrow$ Vector of '1's of length $MaxDT$;
35 for $i$ = $MaxDT$ to 2 do
36     $V_i \leftarrow i^{\text{th}}$ consensus in $ConsVctrs$;
37     for $j$ = ($i$ - 1) to 1 do
38         $V_j \leftarrow j^{\text{th}}$ consensus in $ConsVctrs$;
39         if $Jaccard(V_i, V_j) = 1$ then
40             $ST[i] \leftarrow ST[i] + 1$;
41             Remove $ST[j]$;
42             Remove $V_j$ from $ConsVctrs$;
43         end
44     end
45 end
46 Build the tree of consensuses in $ConsVctrs$
```

Algorithm 1. The MultiCons approach.

Similar consensuses are then removed, and a *stability counter* (ST) is associated to each to count how many times it was generated, i.e., for how many different values of DT (lines 34–45 in Algorithm 1). For example, a consensus with ST = 3 means that this consensus is generated from 3 consecutive values of DT.

Considering the example in Table 1, the first consensus, with DT = 5, consists of instance sets of FCPs number 1 to 4 in Table 2, which FCI size is equal to the number of base clusterings. To build the next consensus, DT is decremented and FCP 5, that defines a pattern common to the 4 base clusterings 1, 2, 4 and 5, is integrated. FCPs 1 and 2 are removed as they are included in FCP 5, and the consensus for DT = 4 thus contains clusters {1,2,3}, {4,5,6,7} and {8,9}. This consensus represents the clustering agreement between at least 4 base clusterings. Since no FCP has an FCI of size 3, the consensus for DT = 3 is identical to the consensus for DT = 4. For DT = 2, FCP 7 is integrated, and replaces FCPs 3 and 5 that are included in FCP 7, resulting in the consensus: {{1,2,3}, {4,5,6,7,8,9}}. For DT = 1, FCP 6 is integrated, and FCP 4 that is included in FCP 6 is deleted. FCPs 6 and 7 are then merged as they intersect, and the consensus for DT = 1 thus results in grouping all instances in 1 cluster, which will become the root of the tree representation described hereafter. In the resulting consensus vector, consensuses for DT = 4 and DT = 3 are merged since they are identical resulting in consensus for DT = 4 having ST = 2. This consensus, that is the most stable generated, represents the best agreement between the 5 base partitions. It tells that at least 4 base clusterings agree on generating its clusters. Actually, the instances of clusters {1,2,3} and {8,9} are grouped together in the 5 base partitions, instances of the first never being grouped with other instances, and the instances of cluster {4,5,6,7} are grouped together in 4 partitions.

But, how can we recognize the best solution in operational situations? As stated before, our objective is to find as many clustering patterns of the hidden cluster structure (or close to it) as possible. This is why, instead of forcibly generate a median partition, we search for clustering patterns from different combinations of base clusterings, and then merge connected ones to build a structure of well separated patterns. However, we can also recommend the best solution as the one that is the most similar to the ensemble using the Jaccard index [10] similarity measure. The consensus with highest average Jaccard similarity with each partition in the ensemble is then recommended as the best solution. In the case of preceding example, this is consensus for DT = 4.

3.4 ConsTree: A Tree of Consensuses

After generating all consensuses, a tree graphical representation is built to visualize the different clustering results: The *ConsTree* tree of consensuses. Each level in the tree depicts a consensus, with nodes representing its clusters and edges representing inclusion relationships between instance sets of clusters of successive levels. The bottom level of the tree is the first consensus.

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's 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's 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* $P_q^{\alpha} \subseteq P_r^{\beta}$*.* L^{α} *is a predecessor of* L^{β} *in the tree of consensuses.*

Figure 1 gives the Hasse diagram of the ConsTree for the example in Table 1. It shows at each level how clusters of the preceding lower level are merged to form new clusters, and the advised level, that is the consensus for DT = 4. We can also note the left branch showing the stable cluster {1,2,3} that is never merged except for root node.

Figure 2 shows another example ConsTree, resulting of the application of MultiCons to a dataset of 399 instances clustered using 10 base clusterings selected randomly, with random settings and K varied from 2 to 11. This tree consists of 7 levels, instead of 10 without merging of identical levels, as duplicated levels for DT = 2 and 3, for DT = 4, 5 and 6 are merged. Visualizing the tree enables the user to understand how the consensuses were built based on different combinations of base clusterings, and to discover strong partitions in the dataset: The cluster(s) that do not merge with others on a sequence of consensuses, which reflect strong intra-cluster similarity between their instances (as the ones circled in blue and red in Fig. 2). It may also highlight groups of instances that are far from being similar to other instances, such as the column of stable cluster circled in red and the similar column beside. Furthermore, the fact that these two columns merge into one cluster, circled in green, rather than merging with any of the previous stable clusters (circled in blue) provides more insight on the peculiar information they hold. The ST value can also point to the result that is more stable than others (the consensus for DT = 6), but as the clustering task is more related to the relevance of the found patterns to user preferences, the user may prefer to select the consensus at DT = 7

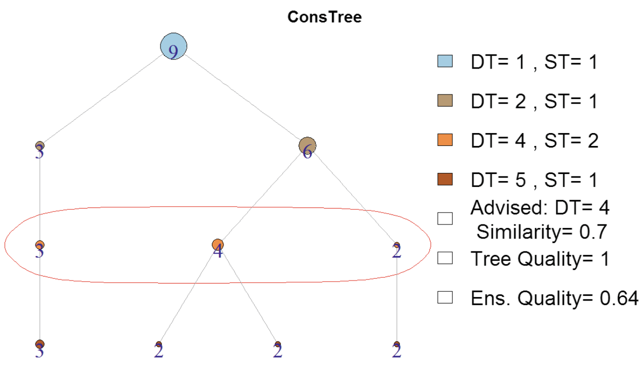

Fig. 1. ConsTree of example membership matrix Table 1.

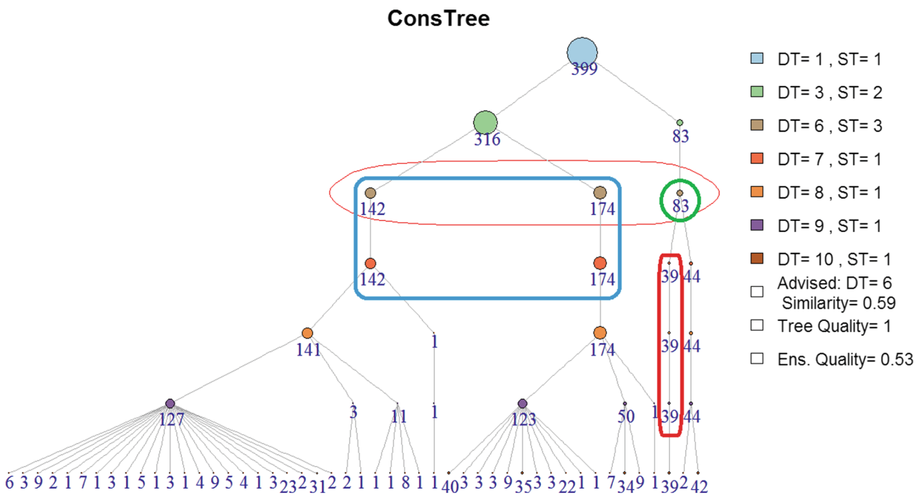

Fig. 2. Example of analysis from ConsTree visualization.

for example, because he/she prefers to separate the cluster of 83 instances at DT = 6 into 2 stable clusters as in DT = 7, as these 2 may reflect better patterns for him/her. Compared to classical ensemble approaches, that only provide one solution representing the clustering that is most similar to the ensemble, the tree of consensuses not only provides more information for the user to understand the strong relations in the data, but it also assists him/her to choose a final clustering based on his/her prior knowledge and preferences.

4 Experiments

We implemented the MultiCons algorithm using R language [16] on a DELL PRECISION M4800 with Intel® Core$^{\text{TM}}$ i7-4710MQ @ 2.50 GHz, 32 GB of RAM, and Microsoft Windows 10 Professional (64-bit) operating system. To generate the frequent closed patterns, we have used the FIST algorithm [13] implemented in Java from the website of the authors.[4] The output of MultiCons is represented as a graph with a vertex for each cluster of a consensus, and an edge between each pair of vertices in two consecutive consensuses representing two clusters related by inclusion. The *plot* function of the *igraph* R package [3] can plot the tree from a data frame defining these edges.

For each test, the base clusterings were generated by random selection of a set of clustering algorithms and/or parameter settings. Among the clustering algorithms used: K-means, PAM, DBSCAN, agglomerative hierarchical clustering, AGNES, DIANA, MCLUST, C-Means, FANNY, Bagged Clustering, and SOM. We compared the results of MultiCons against voting-based consensus

[4] Another possibility is to use the *arules* R package [6].

Table 3. Experimental results.

Dataset	E.Coli	Wine	Zoo	Breast cancer	Smiley	Shapes	Hyper cube	Chain link	Atom	Golf ball	Hepta	Terta
Size	336	178	101	699	500	2000	800	1000	800	4002	212	400
# attributes	7	13	16	9	2	2	3	3	3	3	3	3
# classes	8	3	7	2	4	4	8	2	2	1	7	4
# base clusterings	9	8	18	11	6	8	8	6	8	5	7	7
K range for ensemble	[2, 18]	[2, 10]	[4, 8]	[2, 6]	[2, 7]	[2, 9]	[4, 11]	[2, 7]	[2, 9]	[2, 6]	[4, 10]	[2, 8]
Ensemble min	0.19	0.28	0.29	0.35	0.40	0.46	0.41	0.24	0.46	0.17	0.36	0.50
Ensemble max	0.63	0.87	**0.86**	**0.90**	0.74	**1**	**1**	**1**	**1**	0.50	**1**	**1**
MultiCons	**0.72**	**0.92**	0.78	0.89	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
# clusters in MultiCons	15	4	5	3	4	4	8	2	2	1	7	4
SE	0.43	0.80	0.82	0.89	0.59	0.83	0.89	0.83	0.98	**1**	**1**	**1**
GV1	0.43	0.87	0.79	0.89	0.92	0.66	**1**	0.59	**1**	**1**	**1**	**1**
DWH	0.44	0.70	0.82	0.89	0.56	0.98	0.91	0.53	0.99	**1**	0.78	**1**
HE	0.49	0.89	0.82	0.89	0.78	0.84	**1**	0.64	0.98	**1**	0.78	**1**
SM	0.41	0.85	0.82	0.89	0.75	**1**	0.80	0.65	**1**	**1**	0.98	0.66
GV3	0.49	0.87	0.82	0.89	0.76	**1**	**1**	0.92	**1**	**1**	**1**	**1**
Soft/symdiff	0.36	0.80	0.83	0.89	0.57	**1**	**1**	0.92	**1**	**1**	**1**	**1**
Medoids	0.37	0.87	0.83	**0.90**	0.74	0.87	**1**	0.35	**1**	0.20	**1**	**1**

clustering algorithms available in R package CLUE [8], including the following consensus methods: SE, GV1, DWH, HE, SM, GV3, soft/symdiff, and consensus medoid. To validate the results of our consensus method and the CLUE methods, we compared the clustering results against the true class labels of the tested dataset using several external validation measures like NMI (Normalized Mutual Information) [18], Jaccard, cRand (Corrected Rand), and FM (Fowlkes and Mallows) [4,7] also in R package CLUE [9].

Table 3 presents results of experiments on benchmark datasets, with a summary description of the dataset, the base clusterings, and the quality of the achieved consensus results compared to the true class using Jaccard measure. The Jaccard measure was used because it gives a moderate trade-off between the similarity to the true class and the number of generated clusters [22]. Note that for voting methods in CLUE, all base clusterings in the ensemble must use the same K value. However, we did not impose this constraint in our tests, just set parameter K to the actual K, corresponding to the dataset true classes, for these methods. MultiCons, that does not require parameter K, generates multiple results with different numbers of clusters. The shown result of MultiCons is the one that is the most similar to the ensemble (the recommended consensus).

E.Coli and Wine datasets are available on the UCI Machine Learning Repository [12]. The Zoo, Breast Cancer, Smiley, Shapes and Hyper Cube datasets are available in the *mlbench* R package [14]. The other datasets are from Ultsch [19]. We can see that the MultiCons approach achieved very good results.

5 Conclusions

We presented a new multiple consensus clustering method that does not require parameter setting; yet it can build the appropriate number of clusters based on finding clustering patterns common to the set of base clusterings. This is a major distinction with other consensus methods that require at least parameter K, like CLUE methods, to generate K clusters in the consensus, as without prior domain knowledge, it is difficult to predict K.

To the best of our knowledge, the proposed method is the first to use frequent closed patterns to detect similarities among base clusterings and to build multiple consensuses from these. One of the benefits of using FCIs technique is efficiency: Execution time is not directly related to the size of the dataset, but depends instead on the number of base clusterings used and whether there are many similarities, or many conflicts, between base partitions. Thus, even for large datasets, few FCPs are generated if some clustering patterns are common to most base clusterings, while other methods based on distance matrices are constrained by the size of the dataset. Even when the number of FCPs is large, the greedy processing in MultiCons is fast, since at each consensus, it requires only few FCPs to work with. Tests showed that CLUE methods GV3 and soft/symdiff require a lot of both computation and memory compared to MultiCons for example.

In addition to providing a consensus clustering result, the ConsTree generated by MultiCons serves as a nice data analysis tool. It shows which clusters are stable, which reflect a strong intra-cluster similarity, leading to discovering meaningful patterns in the dataset. On the other hand, stable consensuses (if exist) suggest the existence of strong cluster structure in the dataset, which consists of well-separated clusters. The ConsTree not only provides one solution to the user, but it also allows he/she to choose another solution based on his/her preferences and prior knowledge for separating or merging certain clusters, for instance, the consensus below or above the suggested one.

References

1. Asur, S., Ucar, D., Parthasarathy, S.: An ensemble framework for clustering protein-protein interaction networks. Bioinformatics **23**(13), i29–i40 (2007)
2. Caruana, R., Elhawary, M., Nguyen, N., Smith, C.: Meta clustering. In: Proceedings of the IEEE ICDM Conference, pp. 107–118 (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. Curr. Genomics **10**(6), 430 (2009)
5. Ghaemi, R., Sulaiman, M.N., Ibrahim, H., Mustapha, N.: A survey: clustering ensembles techniques. WASET **50**, 636–645 (2009)
6. Hahsler, M., Gruen, B., Hornik, K.: arules - a computational environment for mining association rules and frequent item sets. J. Stat. Softw. **14**(15), 1–25 (2005)
7. Halkidi, M., Batistakis, Y., Vazirgiannis, M.: On clustering validation techniques. J. Intell. Inf. Syst. **17**(2), 107–145 (2001)
8. Hornik, K.: A CLUE for CLUster Ensembles. J. Stat. Softw. **14**(12), 1–25 (2005)
9. Hornik, K.: CLUE: Cluster ensembles (2015). r package version 0.3-50 http://CRAN.R-project.org/package=clue
10. Jaccard, P.: The distribution of the flora in the alpine zone.1. New Phytol. **11**(2), 37–50 (1912). doi:10.1111/j.1469-8137.1912.tb05611.x
11. Li, T., Ding, C.: Weighted consensus clustering. In: Proceedings of the SIAM Conference on Data Mining, pp. 798–809 (2008)
12. Lichman, M.: UCI machine learning repository (2013). http://archive.ics.uci.edu/ml

13. 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)
14. Newman, D., Hettich, S., Blake, C., Merz, C.: UCI repository of machine learning databases (1998). http://www.ics.uci.edu/~mlearn/MLRepository.html
15. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Efficient mining of association rules using closed itemset lattices. Inf. Syst. **24**(1), 25–46 (1999)
16. R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2015). https://www.R-project.org/
17. 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)
18. Strehl, A., Ghosh, J.: Cluster ensembles - a knowledge reuse framework for combining multiple partitions. JMLR **3**, 583–617 (2003)
19. Ultsch, A.: Clustering with SOM: U*C. In: Proceedings of the WSOM Workshop, pp. 75–82 (2005)
20. Vega-Pons, S., Ruiz-Shulcloper, J.: A survey of clustering ensemble algorithms. IJPRAI **25**(03), 337–372 (2011)
21. 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)
22. Xu, D., Tian, Y.: A comprehensive survey of clustering algorithms. Ann. Data Sci. **2**(2), 165–193 (2015)
23. Yang, G.: The complexity of mining maximal frequent itemsets and maximal frequent patterns. In: ACM SIGKDD, pp. 344–353 (2004)
24. Zhang, Y., Li, T.: Consensus clustering + meta clustering = multiple consensus clustering. In: Proceedings of the FLAIRS Conference (2011)