Browsing Robust Clustering-Alternatives

Martin Hahmann, Dirk Habich, and Wolfgang Lehner

TU Dresden; Database Technology Group; Dresden, Germany
{martin.hahmann, dirk.habich, wolfgang.lehner}@tu-dresden.de

Abstract. In the last years, new clustering approaches utilizing the notion of multiple clusterings have gained attention. Two general directions — each with its individual benefits — are identifiable: (i) extraction of multiple alternative clustering solutions from one dataset and (ii) combination of multiple clusterings of a dataset into one robust consensus-solution. In this paper, we propose a novel hybrid approach to generate and browse robust, alternative clustering results. Our hybrid approach is based on *frequent-groupings* as specialization of frequent-itemset mining. In this way, the different benefits of the existing directions are combined, offering new opportunities for knowledge extraction.

1 Introduction

Depending on which notion you follow, the information age has been around for 20 to 40 years and brought with it a massive trend for digitalization and data collection. Just like coal and steel in the prior age of industrialization, information and data have become a crucial resource for today's work. In contrast to the diminishing natural raw materials, the deposits of data are constantly growing. While data itself is already valuable, it is only capitalized to the full extent if knowledge can be obtained from it. For this task, analysis techniques like clustering have been developed [10]. The goal of clustering is to group a set of data objects into different clusters, so that members of one cluster are similar to each other, while different clusters are dissimilar.

A multitude of clustering algorithms has been proposed over the years [10], whereas these traditional algorithms have several limitations. On the one hand, they are not generally applicable or robust meaning that certain algorithms and parametrizations only suit certain datasets and will yield poor results otherwise. On the other hand, traditional techniques generate only a single clustering, but as todays datasets become more complex and high-dimensional, in general there are more clustering results possible for a dataset. Besides these data centric problems, usability and applicability have become important issues as clustering evolves from a niche application in research to a widespread analysis technique employed in more and more areas. With this trend, new users come into contact with clustering, who are often experts of their respective application domain but have no experience in the area of clustering. This calls for clustering approaches that can be versatilely applied and lack the complicated algorithm selection and configuration of traditional approaches.

In recent years, a number of approaches have been proposed to tackle some of these issues. The area of *alternative clustering* [2, 3] provides multiple clustering solutions for a dataset. With this, several views on complex data can be offered, while the availability of multiple clusterings in the first place, usually frees the user from adjusting a single clustering that proves unsatisfactory. An opposing approach is *ensemble clustering* [11, 4] in which a set of multiple clusterings is integrated to form a single consensus clustering result. This input set is also called clustering-ensemble and contains results that are generated using different algorithms and parameters. The consensus result is often more robust than clusterings generated by a single algorithm and set of parameters, which means that this technique is more versatile in terms of the application scenario. Additionally, algorithm selection and configuration is eased as a range of methods and parameters is utilized. On the downside, ensemble clustering provides just one solution and therefore resembles traditional clustering at that point. To create an alternative clustering solution, the user has to switch and/or re-parametrize the employed clustering algorithms. This is a very challenging task because a set of algorithms must be selected and configured.

To summarize, *alternative clustering* and *ensemble clustering* both have benefits compared to traditional clustering approaches. However, from the user's point of view there is a decision to make. The user must decide if multiple alternative solutions are chosen over one robust solution or vice versa, as it is not possible to have both. In this paper we address this issue by proposing an idea for combining the approaches of alternative and ensemble clustering, that makes the creation of robust alternatives possible. We start with a short description of alternative and ensemble clustering in Section 2. Then, we describe *frequent-groupings* as the core concept our our novel hybrid approach in Section 3. Our *frequent-grouping* technique is based in the idea of frequent-itemset mining [1] and allows the identification of robust clusters, occurring throughout the clustering ensemble. Subsequently, we present how these frequent-groupings can be combined in order to create multiple robust alternatives in Section 4. We outline an algorithm-driven as well as a user-driven approach that enables the creation of alternatives by browsing and switching frequent-groupings. We conclude our paper with a discussion of open issues in Section 5 and a short summary in Section 6.

2 Related Work

The problems of traditional clustering, that we sketched in the introduction often lead to multiple iterations in which different parameters or clustering algorithms are tried out until a satisfactory clustering result is obtained. This trial-and-error practice implicitly generates multiple clustering solutions for the analyzed dataset. Over the last years, new approaches to clustering emerged, that explicitly utilize multiple clustering solutions for the knowledge discovery process. In this section, we briefly review two of these approaches, namely: *alternative clustering*, which provides the user with multiple clustering solutions for a dataset

and *ensemble clustering*, which integrates multiple clusterings of a dataset into a single robust solution.

2.1 Alternative Clustering

The main goal of this clustering approach is to provide alternative clustering solutions to the user. To create alternative solutions, at first an initial clustering result is made, using a traditional clustering algorithm. Based on the information contained in this initial clustering, alternative solutions are generated so that these alternatives are *dissimilar* to the initial solution. As an example for alternative clustering, we describe the *COALA* [2] algorithm. Given a Clustering C with k clusters this method generates a dissimilar alternative S also having k clusters. To express dissimilarity the authors use instance-based 'cannot-link' constraints, that are derived from the initial clustering and are employed in the construction of the alternative. Such a constraint can be expressed as a pair of objects (x_i, x_j) with $i \neq j$. A clustering satisfies this constraint if x_i and x_j are not located in the same cluster. In order to reach the maximum degree of dissimilarity from C, the alternative S should place as much objects as possible in different clusters, that were in the same cluster in C. Although this approach seems plausible, strict adherence to it can lead to meaningless solutions, as a clustering that maximizes dissimilarity most likely does not comply with the general requirements of clustering, namely similar objects belong to the same cluster while clusters are dissimilar. This means besides being dissimilar, an alternative clustering must also satisfy a certain quality, that is, in the case of COALA, expressed by the similarity of a pair of objects. The two goals of dissimilarity and quality can be inversely related, for which case COALA offers a parameter ω to control the trade-off between both. COALA works in an iterative way: the initial clusters contain one object each and are successively merged until k is reached. In each iteration two merge candidates are identified: one cluster pair that pursues the quality goal by having the smallest distance (d_{qual}) of all pairs and one cluster pair pursuing dissimilarity, that has the smallest distance (d_{diss}) of all pairs that fulfill the cannot-link constraints. The decision between both candidates is based on an inequation: if $d_{qual} \leq \omega \cdot d_{diss}$ the quality pair is merged, else the dissimilarity pair. Another alternative clustering technique is CAMI[3] which is also based on the goals of quality and dissimilarity but models them in a different way as it represents clusters using gaussian mixtures.

2.2 Ensemble Clustering

In contrast to alternative clustering, ensemble clustering does not present multiple clusterings to the user, but uses them to generate a single clustering result. This set of multiple clusterings is also called *clustering-ensemble* and contains clustering results generated by executing multiple algorithms with multiple sets of parameters. As is known, certain algorithms or parametrizations do not suit certain datasets, thus producing poor results. By utilizing a wide range of different clustering algorithms and parametrizations, this problem can be tackled.

Thus, the final clustering solution—called *consensus* clustering—generated from such a clustering ensemble is more robust and often has an increased quality [11]. In order to create such a clustering a consensus function is needed, that integrates the cluster assignments of all ensemble members into a new clustering. The use of the term consensus already shows that the goal of this integration is to identify clusters/structures that are detected by the majority of ensemble members and preserve them in the final solution. In other words, similarities throughout the clustering-ensemble are identified and used to create the consensus clustering. Two main classes of ensemble clustering techniques can be distinguished: *pairwise-similarity* approaches and approaches based on *cluster-labels*.

Pairwise-Similarities: Algorithms working on the basis of pairwise similarities model the cluster assignment information by evaluating the assignments of each object pair over the whole ensemble [4, 11]. There are two cases of pairwise similarity: (i) a pair objects is part of the *same* cluster or (ii) a pair of objects is part of *different* clusters. For each local clustering of the ensemble these similarities can be represented in the form of a so called coassociation matrix, in which a cell contains a 1 if the respective pair of objects is located in the same cluster or a 0 if an object pair is assigned to different clusters. By adding up all the local matrices and normalizing each cell using the ensemble size, a global coassociation matrix is build that contains the relative frequency in which each pair of objects is located in the same cluster throughout the whole ensemble. Based on this matrix, different consensus functions can be employed to extract the final solution. As an example. we describe a very simple function based on [4], which generates a consensus clustering on the following basis: if a pair of objects is located in the same cluster in the majority of the ensemble, i.e. at least 50% of the clusterings, it should also be part of the same cluster in the consensus solution. Vice versa this also holds for object pairs mostly located in different clusters. Therefore, the consensus clustering shows minimal dissimilarities to the ensemble in terms of pairwise similarities. The consensus function is realized by removing all cells from the global coassociation matrix that contain a value smaller than 0.5 and use the remaining cells to generate the clustering.

Cluster-Labels: In contrast to pairwise-similarity based approaches, other techniques for ensemble clustering directly use the cluster assignments from the ensemble by working with the provided cluster labels only. As no coassociation matrices are used, they are often less time-consuming. Various algorithms exist in this class e.g. *Ensemble-Merging* [9] which assumes a clustering ensemble having a constant number of clusters k, that are each represented by a centroid. By grouping similar centroids of the ensemble, k global centroids are generated which are then used to build the consensus clustering.

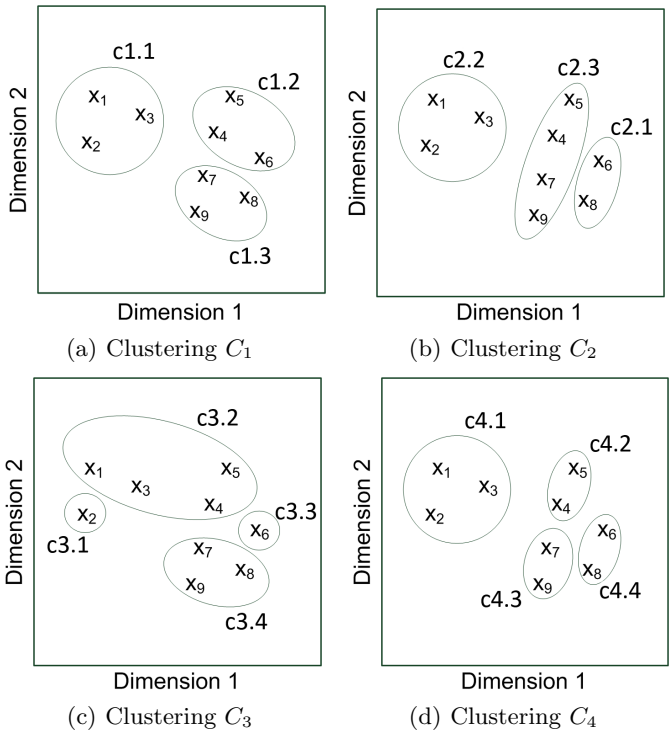
3 Finding Frequent Groupings

In the previous section, we described that ensemble clustering creates a single consensus clustering out of a set of different multiple clusterings. We outlined the two major approaches to this task, namely cluster-labels and pairwise-

similarities. While the techniques of both approaches differ in their mode of operation, they share the common goal to incorporate those parts of the dataset into the final solution, whose cluster assignments agree with the majority of the clustering-ensemble. In other words, clusters of the consensus solution are sets of objects, that were frequently assigned to the same cluster throughout the clustering-ensemble. For the purpose of illustration, we use the small clustering-ensemble depicted in Figure 1 as a running example. Our example contains a dataset $\mathcal{D} = \{x_1, x_2, \ldots, x_9\}$ of nine objects in a $2D$ feature space. For $\mathcal{D}$ exists a clustering-ensemble $\mathcal{C} = \{C_1, C_2, C_3, C_4\}$ that contains four clusterings, each with a different number of clusters and different cluster composition. To model and evaluate the similarities of the cluster assignments of an object in the ensemble, label-based approaches match cluster ids or representations, while approaches based on pairwise-similarities count the co-occurence of object pairs. To give an example for a consensus clustering, we apply the pairwise approach mentioned in Section 2 to $\mathcal{C}$ and obtain a clustering with the clusters $c_1 = \{x_1, x_2, x_3\}$, $c_2 = \{x_4, x_5\}$, and $c_3 = \{x_6, x_7, x_8, x_9\}$. In addition to these two principles, we propose a novel third way to identify the frequent assignment of objects to the same cluster, which is based on the concept of frequent itemsets [1].

Assuming a set of n items $\mathcal{I} = \{i_1, i_2, \ldots, i_n\}$ and a set of transactions $\mathcal{T} = \{t_1, \ldots, t_m\}$ of which each transaction has a unique id an contains a subset of $\mathcal{I}$, a set of items $\mathcal{X}$ is called frequent if its *support* exceeds a given threshold. The support of an itemset $\mathcal{X}$ is defined by the fraction of transactions of $\mathcal{T}$ that contain $\mathcal{X}$. At this point, the analogy to ensemble clustering should be obvious: while frequent itemsets aim to identify items that co-occur in many transitions while ensemble clustering searches for objects occuring together in the majority of clusters. In the following, we map the concepts of $\mathcal{I}$, $\mathcal{T}$ and *support* to the domain of ensemble clustering in order to describe a method that allows the identification of *frequent-groupings*.

While it is obvious that the items of $\mathcal{I}$ correspond to the objects of the dataset for a clustering, the matching of the transaction concept is more intricate. As $\mathcal{T}$ is a set of transactions, that each contain elements of $\mathcal{I}$. At first sight, it could be mapped to the clustering-ensemble, because the ensemble also consists of multiple clusterings, containing the elements of the dataset. However this mapping should not be used, because it effectively prevents the identification of frequent sets of objects, as in general, all objects of a dataset are assigned to a cluster. Assuming the transaction-clustering analogy, this means that each transaction contains all items of $\mathcal{I}$, which makes it impossible to identify interesting, frequent object groupings. Therefore we model $\mathcal{T}$ as the set of clusters from all members of the clustering-ensemble, as each of them normally only contains a subset of the data. Using the mappings made so far, the *support* of a set of data objects $\mathcal{X}$ shows the fraction of the clustering-ensemble, in which $\mathcal{X}$ is part of the same cluster. If $support(\mathcal{X})$ exceeds a certain threshold, we call $\mathcal{X}$ a *frequent-grouping*. A high support of $\mathcal{X}$ also shows that this set of objects is robust, because it was



(a) Clustering C_1 (b) Clustering C_2

(c) Clustering C_3 (d) Clustering C_4

Fig. 1. The clustering-ensemble of the running example.

identified as part of a cluster in many clusterings, regardless of the employed algorithm and/or parameters.

With regard to our running example, this means that $\mathcal{D}$ acts as itemset, while the clustering-ensemble $\mathcal{C}$ provides a set of 14 clusters/transactions. As transactions are required to have unique identifiers, we label them in a special way e.g. $c1.2$ marks cluster 2 of clustering C_1. To simplify the calculation of support, we make the following assumption: each $x_i \in \mathcal{D}$ is assigned to exactly one cluster in each $C_i \in \mathcal{C}$. With this, an object can only occur in one cluster resp. transaction per clustering. To illustrate what this means, we regard the group of objects (x_1, x_2, x_3) from our running example. These objects occur in the three clusters $c1.1, c2.2$, and $c4.1$ thus this itemset has a support of $3/4$ resp. 0.75 as it occurs in three clusterings C_1, C_2, and C_4.

In order to generate the frequent-groupings from our running example, we first must specify a threshold for the support to decide whether a group of objects occurs frequently or not. Following the assumptions of [4] we regard a set of objects as frequent, if it occurs at least in 50% of the clustering-ensemble, which means two clusterings in our example. The obtained frequent-groupings are depicted in Figure 2 in the form of a graph structure. Each node represents a

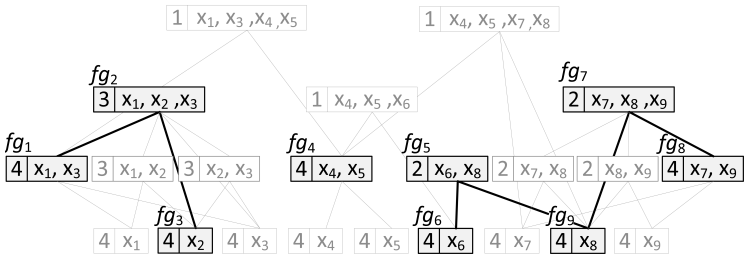


Fig. 2. Structure of the frequent-groupings generated from the running example.

frequent-grouping and contains its associated objects as well as its support, while edges indicate subset/superset relations between nodes. To build this structure we initially create and insert a node for each cluster found in the ensemble. If a node already exists in the graph structure it is not inserted again but the support of the existing node is increased by one e.g. the objects $\{x_1, x_2, x_3\}$ are contained in the clusters $c1.1, c2.2$, and $c4.1$, thus only one node with a support of three is generated. From the initial graph all nodes that are not frequent, i.e. with a support less than two are filtered—e.g. $c3.2$—and are displayed in a faded grey. For each remaining frequent-grouping a new set of nodes is created, containing all of its possible subsets. At last all frequent-grouping nodes that are not *closed* are filtered i.e. each node, having a direct superset with the same support is removed from the graph. Eventually this procedure leads to the nine frequent-groupings $fg_1, fg_2, \ldots, fg_9$ displayed in Figure 2. Please note that the described procedure only illustrates the formation of the depicted graph structure. There already exist sophisticated methods for mining closed frequent itemsets, that can be applied to generate frequent-groupings more efficiently [12].

By interpreting the obtained frequent-groupings as clusters, we can generate a consensus clustering by combining them in a way, that all objects of $\mathcal{D}$ are located in a cluster. As the frequent-groupings overlap, multiple *alternative* combinations can be produced. For our running example, six alternative consensus clusterings can be created, which are shown in Table 1. We will discuss the actual construction of these alternatives in the following section. In contrast to the alternative clusterings generated by existing approaches, our solutions feature a certain degree of robustness, as for each cluster a consensus exists throughout the ensemble, which is defined via the support threshold. Thus our frequent-grouping approach represents a hybrid between alternative and ensemble clustering, that combines the benefits of both domains, namely alternative solutions and robustness. In addition our approach has some advantages over the cluster-label and pairwise-similarity based techniques. Most label based approaches require that the number of clusters in the consensus solution is specified in advance. This is not necessary with pairwise-similarities or frequent-groupings as the number of consensus clusters results from the co-occurrence of objects in the ensemble. Although this characteristic is a similarity between both techniques there is a difference. As pairwise-similarity methods work with the smallest possible group-

A_1	$\{x_1, x_2, x_3\}\,\{x_4, x_5\}\,\{x_6\}\,\{x_7, x_8, x_9\}$
A_2	$\{x_1, x_2, x_3\}\,\{x_4, x_5\}\,\{x_6, x_8\}\,\{x_7, x_9\}$
A_3	$\{x_1, x_2, x_3\}\,\{x_4, x_5\}\,\{x_6\}\,\{x_8\}\,\{x_7, x_9\}$
A_4	$\{x_1, x_3\}\,\{x_2\}\,\{x_4, x_5\}\,\{x_6\}\,\{x_7, x_8, x_9\}$
A_5	$\{x_1, x_3\}\,\{x_2\}\,\{x_4, x_5\}\,\{x_6, x_8\}\,\{x_7, x_9\}$
A_6	$\{x_1, x_3\}\,\{x_2\}\,\{x_4, x_5\}\,\{x_6\}\,\{x_8\}\,\{x_7, x_9\}$

Table 1. Alternative consensus clusterings of the running example.

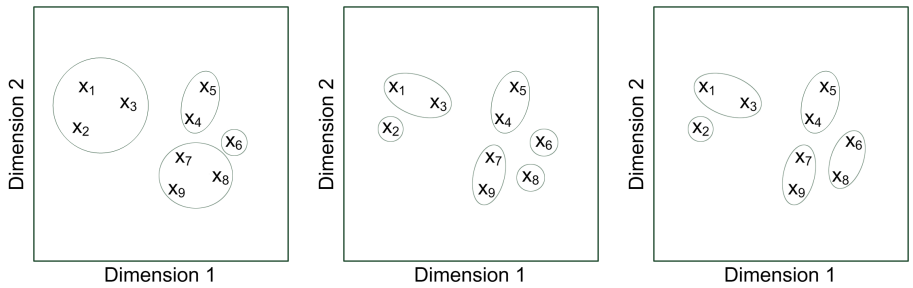
ings i.e. pairs, they are prone to transitive effects. Assume two object pairs (p, q) and (q, r), each one occuring in the same cluster in 50% of the ensemble. Such a setting can lead to (p, q, r) being placed in the same cluster of the consensus solution, even if (p, r) never occurs in the same cluster in the whole ensemble. This cannot happen with frequent-groupings as it evaluates co-occurrence in a different way.

4 Browsing Alternatives

Aside from being able to identify frequent-groupings, we need a combination procedure to generate alternative solutions. The challenging part here is the high combinatorial diversity resulting from the concept of frequent-itemsets and our idea of combination. To illustrate this issue we again look at our running example $\mathcal{D}$, which contains nine objects. All possible frequent-groupings that can occur in a dataset D are contained in the *power set* of D. As we do not need the empty set, there are $2^{|D|} - 1$ possible groupings—i.e. 511 for our example—to begin with. This extremely high number indicates the general scale, we are dealing with but has no direct impact on our approach, as we only consider the groupings that were found in the ensemble and their respective subsets. Besides the reduction by clustering, further candidates for frequent-groupings are removed via the support threshold and the condition that frequent-groupings must be closed. Notably this last filter criterion is important as it keeps the number of small and singleton frequent-groupings low. These small groupings inflate the number of possible alternative consensus clusterings by allowing additional combinations that differ only in the assignment of one or two objects. For our running example this is neglectable but for larger data, this issue must be considered i.e. too small frequent-groupings must be removed. In section 3, we have assumed that each object is assigned to one cluster in each clustering of the ensemble, which means that potentially $|D|$ singleton groupings with a support of 100% exist, each containing one element of D. If these are not considered, the number of alternative solutions rises and even the trivial solution of a clustering that assigns each object to its own cluster is possible.

Having discussed the necessity of frequent-grouping filtering we are now faced with the question of obtaining different alternative clustering solutions. To get all possible alternatives it would be necessary to generate all possible combinations of frequent-groupings—the power set of the set of frequent-groupings—and select

all combinations that contain D and only consist of disjoint frequent-groupings. Obviously this approach is very expensive, thus we propose the use of a greedy approach for the construction of alternatives. By varying the optimization criterion a set of alternatives can be generated. Using the frequent-groupings shown in Figure 2, we extracted the three alternatives depicted in Figure 3 by using three different selection criterions. The alternative in Figure 3(a) was generated with the goal of maximizing cluster size, therefore it contains the frequent-groupings fg_2, fg_4, fg_6, fg_7 and matches alternative A_1 from Table 1. Furthermore, alternative A_6 of Figure 3(b) maximizes the support, while A_5 of Figure 3(c) aims to maximize the number of equal-sized clusters. Naturally it is possible to employ other optimization goals than frequent-grouping size and support. The identification of such goals and more complex greedy heuristics is a topic for future research.



(a) A_1: maximize cluster size (b) A_6: maximize support (c) A_5: equal cluster size

Fig. 3. Alternative consensus clusterings extracted with different greedy approaches.

Using greedy approaches to extract alternatives means that the user needs to specify the number of alternatives, a number of greedy heuristics, and maybe additional parameters. Regarding different levels of user experience and the characteristics of different application domains, this can be a challenging task. Therefore, we propose a second way to generate robust alternatives that allows users to actually browse through alternatives using high-level feedback. This approach relies on our previous work described in [5], where we propose a feedback-driven process for ensemble clustering that allows users to iteratively refine a consensus clustering using a visual-interactive interface [6, 7] and a special paiwise-similarity based ensemble clustering approach [8]. Our proposed process provides an initial clustering solution, which the user interpretes in terms of intra-cluster composition and inter-cluster relations via the proposed visual-interactive interface. Depending on the users evaluation, the initial result can be adjusted using a set of four feedback operations: *merge, split, refine, and restructure*. In the following we transfer the general idea of this process and its feedback options to the setting of our frequent-groupings approach. Therefore we assume

that the user is provided with an initial alternative, created by an arbitrary greedy approach. Furthermore we assume that access to some sort of clustering visualization is available. In this setting the user can now use the four feedback operations to browse through the structure given by our frequent-groupings and their subset/superset relations. In doing so, the user can create different robust alternative clusterings. Subsequently, we use our running example to illustrate this feedback-driven browsing and describe the implementation of our feedback operations in this context. An overview of the implementation is depicted in Figure 4. As intial consensus clustering, we assume alternative A_5 shown in Figure 3(c). Should the user not be satisfied with the clusters fg_1, fg_3 it is possible to combine them into one cluster by applying the *merge* feedback operation to both. To realize the merge, the superset relations resp. the ascending edges of fg_1 and fg_3 are checked inside the graph structure. If these relations meet in a frequent-grouping that is a superset of the originating clusters and equals their union, both original clusters are replaced by the new found superset. In our running example this requirement is fulfilled, thus fg_1 and fg_3 are replaced by fg_2 which effectively transforms A_5 into A_2. By issuing this simple operation the user has generated a new robust alternative. If there exists no suitable superset, then *merge* is not applicable. Based on this, the *split* operation is implemented by traversing the subsets resp. the descending edges of a frequent-grouping towards the nearest set of frequent-groupings, that are subsets of the original grouping and exactly contain all of its members. In our example, the *split* of fg_5 would replace this frequent-grouping with fg_6 and fg_8, thus transforming A_5 into A_6. Again, if there are no suitable subsets—e.g. for fg_4—then a split is not possible. By analyzing the frequent-groupings in advance, availability of split an merge operations can be determined for each frequent-grouping and displayed in the visualization.

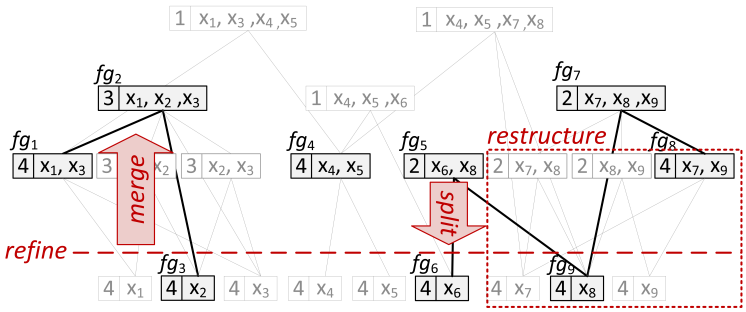


Fig. 4. Implementation of high-level feedback for browsing frequent-groupings.

The remaining feedback operations *refine* and *restructure* are always applicable as they do not navigate through the frequent-groupings but are used to remove or re-create them. With *refine*, frequent-groupings that do not exceed a

certain size are filtered. This allows the removal of clusters considered too small, in the respective application domain and the already mentioned singletons, thus reducing the number of available alternative consensus clusterings. Applying refine in order to eliminate singletons in our running example results in the pruning of fg_3, fg_6, and fg_9, and thus would reduce the available alternatives to A_2. Although this seems like a harsh reduction in our small scale example this method has its merits for larger datasets. Furthermore, use of the refine operation can result in objects of D not being covered by any frequent-grouping. In this case these objects can be considered as *noise* in the consensus clustering, as they cannot be assigned to a robust cluster of significant size. This notion of noise also influences the number of available alternatives which we will demonstrate using the already mentioned application of refine to the running example. If we rule out the existence of noise, the only possible consensus clustering is A_2. On the other hand, the existence of noise makes new alternatives possible, for example A_1 would become $\{x_1, x_2, x_3\}\{x_4, x_5\}\{x_7, x_8, x_9\}$ with x_6 being noise, thus presenting a clustering solution that not only contains robust clusters but also identifies those parts of the dataset for which satisfactory consensus cannot be found. Based on this, it could be possible to draw conclusions regarding dataset structure and pre-processing. Therefore the handling and implications of this kind of noise will be part of our future research.

At last, *restructure* allows the reconfiguration of certain frequent-groupings. Lets assume that a user is especially interested in a specific area of the dataset, but the available frequent-groupings provide no or not enough alternatives resp. split/merge possibilities for this area. In this case *restructure* builds a new clustering-ensemble and new frequent-groupings for the specified subset of D. Application of this operation to fg_7 means for example, that fg_7, fg_8, and fg_9 are replaced by the frequent-groupings, resulting from a new clustering-ensemble generated for $\{x_7, x_8, x_9\}$. The restructure operation should only be used if, based on domain knowledge, other frequent-groupings can be expected in the respective area.

5 Open Issues

Regarding the generation of frequent-groupings, existing algorithms for frequent itemset mining should be adapted to this domain, focusing especially on pruning techniques, measures for interestingness and the influence of different support thresholds. Other areas of interest are the creation of frequent-groupings from fuzzy clustering-ensembles and more generally the evaluation of support without the assumptions, that an object is always assigned to exactly one cluster in each clustering.

For the automatic extraction of robust alternatives, it is necessary to develop additional greedy heuristics. These heuristics should incorporate the notions of quality and dissimilarity that are found in many existing alternative-clustering approaches into the extraction process in an advatageous way. As frequent-groupings and their superset/subset relations can be interpreted as nodes and

edges of a graph, methods for graph-partitioning also promise to be a viable option for the creation of alternative clustering solutions. Additionally, the notion of noise introduced in the previous section must be explored further. On the one hand, methods for filtering too small frequnet-groupings must be developed in order to keep the number of robust alternative clusterings manageable. On the other hand, the meaning of this noise i.e. of objects for which no satisfying consensus can be found, needs to be explored further, as this could allow conclusions regarding the pre-processing of the data or the fit between dataset and employed algorithms/parameters.

Regarding the proposed user-driven browsing of robust alternatives there are open issues like the integration of specific information like support or relations between frequent-groupings into our visual-interactive interface. Furthermore it is necessary to find a way to handle large numbers of alternatives i.e. methods for communicating the availability of many alternatives to the user must be found. In additon the stepping for navigating through alternatives must be chosen adequately. We are positive that our frequent-groupings approach offers many additional opportunities for further research.

6 Summary

In this paper we proposed our idea of combining the concepts of alternative and ensemble-clustering. Based on the well-known technique of frequent itemset mining, we introduced our notion of frequent-clusters as robust/frequently occuring parts of the clustering ensemble. After describing the creation process of frequent-clusters, we proposed an algorithmic and an user-driven approach for the construction of alternative consensus-clusterings from frequent-clusters. The algorithmic approach uses greedy extraction methods and different optimization goals and thus needs to be parametrized by the user. The user-driven browsing on the other hand employs a small set of high-level feedback options, that allows users to navigate and adjust frequent-clusters in order to compose different consensus-clusterings. The clustering results that are obtained with our approach combine benefits from the domains of alternative and ensemble-clustering, namely multiple alternative solutions that are robust.

References

1. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. pages 487–499, 1994.
2. E. Bae and J. Bailey. Coala: A novel approach for the extraction of an alternate clustering of high quality and high dissimilarity. In *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006)*, pages 53–62, 2006.
3. X. H. Dang and J. Bailey. Generation of alternative clusterings using the cami approach. In *Proceedings of the Tenth SIAM International Conference on Data Mining*, pages 118–129, 2010.
4. A. Gionis, H. Mannila, and P. Tsaparas. Clustering aggregation. In *Proc. of ICDE*, 2005.

5. M. Hahmann, D. Habich, and W. Lehner. Evolving ensemble-clustering to a feedback-driven process. In *Proceedings of the IEEE ICDM Workshop on Visual Analytics and Knowledge Discovery (VAKD)*, 2010.
6. M. Hahmann, D. Habich, and W. Lehner. Visual decision support for ensemble-clustering. In *Proceedings of the 22nd International Conference on Scientific and Statistical Database Management (SSDBM)*, 2010. (to appear).
7. M. Hahmann, D. Habich, and W. Lehner. Touch it, mine it, view it, shape it. In *Proceedings der 14. GI-Fachtagung für Datenbanksysteme in Business, Technology und Web (BTW 2011, February 28 - March 4 2011, Kaiserslautern, Germany)*, 2011.
8. M. Hahmann, P. Volk, F. Rosenthal, D. Habich, and W. Lehner. How to control clustering results? flexible clustering aggregation. In *Advances in Intelligent Data Analysis VIII*, pages 59–70, 2009.
9. P. Hore, L. Hall, and D. Goldgof. A cluster ensemble framework for large data sets. In *IEEE International Conference on Systems, Man, and Cybernetics*, 2006.
10. A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3), 1999.
11. A. Strehl and J. Ghosh. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3, 2002.
12. M. J. Zaki and C. jui Hsiao. Charm: An efficient algorithm for closed itemset mining. pages 457–473, 2002.