Anchored Constrained Clustering Ensemble

Mathieu Guilbert, Christel Vrain, Thi-Bich-Hanh Dao, Marcilio de Souto

Anchored Constrained Clustering Ensemble

Mathieu Guilbert, Christel Vrain, Thi-Bich-Hanh Dao, Marcilio C. P. de Souto
Univ. Orléans, INSA Centre Val de Loire, LIFO EA 4022
Orléans, France
{mathieu.guilbert, christel.vrain, thi-bich-hanh.dao, marcilio.desouto}@univ-orleans.fr

***Abstract*—In the context of semi-supervised learning and clustering ensemble methods, we introduce a novel strategy to consider not only pairwise constraints, but also triplet constraints. As far as we are aware, the latter have not been addressed in the literature of semi-supervised clustering ensembles. The strategy consists of a post-processing applied once a consensus partition has been built. Taking into account the fact that the clusters of the consensus partition are usually not spherical, in order to maintain their complex shapes, we first generate anchors, which are data points judged representative of the clusters, in such a way that every point in the cluster has an anchor close to it. These anchors are then used to create a data structure that we call an allocation matrix, which measures the assignment score of each point to each cluster in the consensus partition. Such a matrix is provided to an Integer Linear Programing (ILP) model to find the partition which is closest to the consensus partition, while satisfying the constraints. The experimental results show that our method, given an initial consensus partition and a set of constraints, allows to satisfy all the given constraints, modifying the initial partition, but without deteriorating considerably its quality.**

***Index Terms*—clustering, clustering ensemble, Integer Linear Programming**

I. Introduction

Clustering is a machine learning task that aims at categorizing unlabeled instances of a dataset into classes (or clusters). Elements within one cluster should be similar to each other while being dissimilar to those of other clusters. The result of a clustering process can take different forms such as a hard/soft partition or a hierarchy of partitions. In this paper we will consider the case where the output is a hard partition. In Semi-Supervised clustering, prior knowledge in the form of constraints is added in order to guide the process towards a partition close to the user expectations. The most common constraints are pairwise instance-level constraints as, for example, *must-link (ML)* and *cannot-link (CL)*. These types of constraints indicate that two instances must be (respectively cannot be) in the same cluster in the final partition. Many clustering algorithms, such as K-Means [1] assume that the clusters are spherical and search for compact and well separated clusters. Some methods have been designed to overcome this limitation, such as density-based methods, as for instance DBScan [2] or clustering ensemble methods that take as input a set of partitions generated with different techniques or different parameters and build a consensus partition.

Semi-Supervised Clustering Ensemble (SSCE) is a growing area of research, which aims at integrating constraints in clustering ensemble methods [3]. The existing approaches consider only pairwise constraints at different stages of the ensemble clustering process. These approaches mainly focus on pairwise constraints, whereas it has been shown that other kinds of constraints could be integrated [4], for instance cardinality constraints on the clusters or even more semantic constraints on some expected properties of individuals in a cluster. Moreover, it has also been shown that declarative frameworks such as Integer Linear Programming (ILP) [5], SAT [6], Constraint Programming [7] allow to easily model and integrate such constraints in a clustering process.

Being able to satisfy the constraints given by an expert is a key point in a exploratory data mining process, all the more when the expert is integrated in the discovery process (*Human in the Loop*). In this paper, we propose a new method to take into account the constraints in a clustering ensemble method. Once the consensus partition has been generated, an ILP model [8] allows to slightly modify it in order to satisfy the constraints. It requires the building of an allocation matrix that gives a score for a point to belong to a cluster. This approach was originally developed with spherical clusters in mind, as the allocation matrix was built from the centroids of the clusters. In this work, we aim at using it on arbitrary shaped clusters. To do so, we introduce the concept on anchors that are instances that can be seen as representatives of their neighborhoods, and the allocation matrix is built on the distances of points to their nearest anchors. Once the matrix is built, the ILP model finds an assignment of points to clusters maximizing the score of each instance assignment while satisfying all the constraints.

Our main contributions are:

- a new method for handling constraints in ensemble-clustering, based on a post-processing of the consensus partition,
- as it is based on declarative frameworks, it can be applied to different kinds of constraints, beyond the classic pairwise constraints,
- it can be combined with other strategy to handle constraints (in base partitions or when building a co-association matrix, ...): this is illustrated in the experiments, where base partitions are generated by COP-Kmeans,
- experiments are performed on pairwise constraints and triplet constraints, showing that our method allows to satisfy all the given constraints without deteriorating significantly the quality of the consensus partition.

The paper is organized as follows: Section II presents the

related work; in Section III we introduce our approach while Section IV presents our experimental protocol and results; finally, in Section V we highlight our main findings and discuss future works.

II. State of the art

A clustering ensemble method starts by the generation of base partitions that will be used to build the final partition. The initial partitions obtained in this step determine the final result, and the generation process of these partitions is thus very important. Different clustering algorithms can be used, or the same algorithm with different parameters, or even different data representations, different subsets of objects or even projections of the objects on different subspaces.

The consensus function is the main step in any clustering ensemble algorithm. In this step, the final data partition or consensus partition P^* is computed. There are two main methods for computing it, either based on the building of a median partition or on the object co-occurrences. In this paper, we only focus on approaches based on co-occurrences.

One of the most classical co-occurrence methods is called Evidence Accumulation (EAC) [9]. It is based on the construction of a co-association matrix. The idea is to combine the results of multiple partitions into a single data partition by viewing each clustering result as an independent evidence of data organization without any assumptions on the number of clusters either in the base partitions or in the consensus partition. More precisely, in a co-association matrix S, S_{ij} will be the number of base partitions where instances i and j are clustered together divided by the number of base partitions. Such a matrix can be seen as a similarity matrix. Finally, to obtain the consensus partition, any clustering algorithm can be applied on the co-association matrix.

In recent years, Constrained Clustering that aims at integrating expert knowledge in a clustering process has become an increasingly active domain of research [10], [11].

Some theoretical studies have shown that very accurate predictions can be obtained by Semi-Supervised Clustering Ensemble Methods [12] and a lot of approaches have been introduced since around 2010. The constraints can be introduced in different steps, which are not mutually exclusive: by reducing the dimensions [13], during the creation of the base partitions [14]–[17], through a partition selection process [14], [18], for the creation of the co-association matrix [13] or in the consensus function [19]. These approaches integrate only must-link and cannot-link constraints.

On the other hand, some work on constrained clustering have shown the interest of declarative frameworks for modeling a large diversity of constraints, such as cardinality constraints, geometric constraints (for instance on the diameter of clusters or on the split between clusters), density constraints, ...They are based on SAT [6], ILP [5] or Constraint Programming [4], [7]. In our work, we investigate the integration of must-link, cannot-link and triplet constraints in clustering ensemble. A triplet constraint (a, p, n) indicates that a is closer to p than to n, that can be interpreted as follows: if a and n are assigned to the same cluster than p must be also. In our knowledge, this constraint type has not been integrated in semi-supervised clustering ensemble.

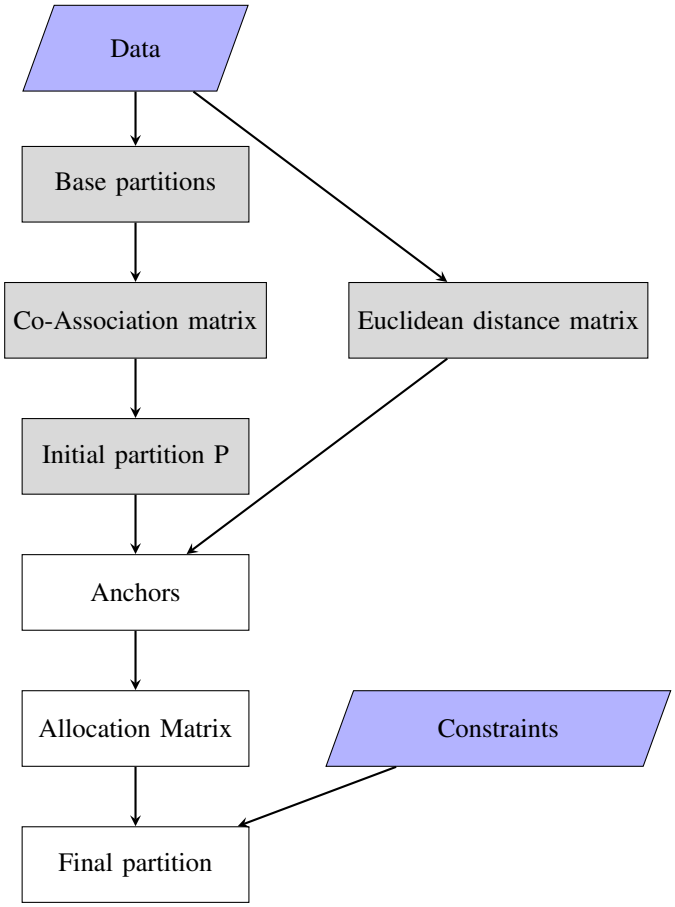


Fig. 1: General Approach
In blue: inputs. In gray: initialization. In white: elements of our own method.

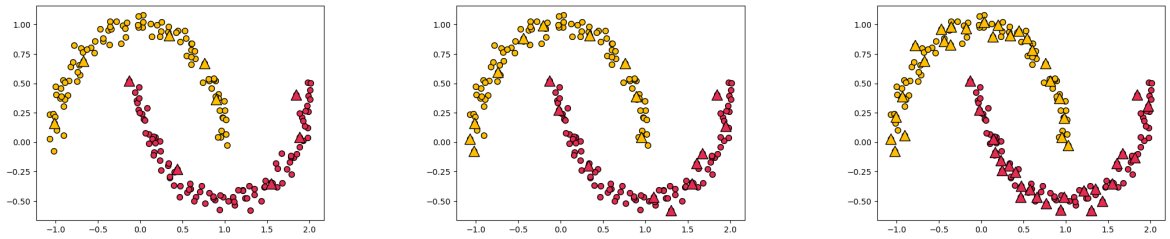
III. Anchored Constrained Clustering Ensemble (ACCE)

A. Overview of our approach

We propose a new method called Anchored Constrained Clustering Ensemble (ACCE) as described in Fig. 1, which integrates the constraints thanks to a post-processing approach.

Given a dataset, an ensemble of base partitions is generated and a consensus partition P^* is built, by applying the Evidence Accumulation algorithm (EAC) [9] with the Single link algorithm to generate the consensus partition.

We then build an allocation matrix M between data points and clusters that gives to each point a score for belonging to each cluster. Since the shape of the clusters could be non-spherical, this score cannot be computed from the distance of a point to the center of its cluster. Therefore for each cluster of the consensus partition, we generate a set of anchors to represent the cluster, and the score for any point i to any cluster c will be computed through the closest anchor of the



(a) 5% of anchors (b) 10% of anchors (c) 25% of anchors

Fig. 2: Anchor positions with different parameters
The anchors are represented by triangles

cluster c to the point i. The choice of the anchors is a key point in our work since they define the way the allocation matrix is built.

Finally, we apply an ILP algorithm with the Allocation Matrix M and the set of given constraints to obtain the final partition. It relies on an optimization criterion aiming at maximizing the agreement between the information in M and the result.

B. Anchor computation

Given a partition of a dataset, anchors are points judged as representatives of their neighborhood. A set of anchor points is computed for each cluster of the partition. Their purpose is to help the construction of the allocation matrix.

The algorithm computing anchors is given in Algorithm 1. One of the required parameters is the percentage per of points that will become anchors in each cluster. For each cluster in the consensus partition P^* we apply the single link hierarchical algorithm to divide it into h smaller sub-clusters, where h is the number of anchors to generate according to the parameter $per \in \{0, ..., 100\}$. For each of these sub-clusters we define an anchor as the instance minimizing the sum of its distances with the other instances of the sub-cluster.

Fig. 2 shows the position of anchors in the dataset `halfmoon`. The two clusters have different colors and the anchors are represented by triangles. We can observe that the anchor distribution is not homogeneous; this is due to the utilization of the Single-Link algorithm. The heterogeneity varies according to the datasets and algorithms applied. However, it still provides a better coverage of the clusters than just using their centroids as it is the case in [8].

Algorithm 1: AnchorComputation

Data: Euclidean distance matrix $distMat$, number of clusters K, percentage per, partition P^*
Result: $ListAnchor$ a list of anchors

```
begin
    for each cluster clustK ∈ P* do
        //Defining h the number of subclusters in clustK
        if clustK has more than 1 point then
            h = |clustK| * per/100
        else
            h = 1
        P_h = result of Single-link with h clusters on clustK
        for each cluster C ∈ P_h do
            anchor = arg min_{u∈C} Σ_{v∈C} distMat(u, v)
            Add anchor to ListAnchor
    return ListAnchor
```

C. Allocation Matrix

Given a partition of N points to K clusters (e.g., the initial consensus partition), we aim at modifying this partition, so as to satisfy the constraints. We introduce the allocation matrix M, a matrix of size $N \times K$ where M_{ik} represents the score associated if the i-th point were assigned to cluster k.

The algorithm to build this matrix is described in Algorithm 2. The first step computes a matrix D where D_{ik} is the distance between i and the closest anchor belonging to cluster k. The farthest the point to its anchor, the smallest its score. We first introduce a matrix R where R_{ik} is the normalization of D_{ik} by the sum of the distances in D_i. We then transform distances into scores by taking $1 - R_{ik}$. Finally, to obtain the matrix M, we re-normalize the result by dividing the scores by $K - 1$, so that each row sums up to one.

D. Integer Linear Programming model

The final step of our approach is the use of an ILP method first introduced in [8]. This method takes as input an allocation matrix M on N instances to K clusters, and finds an assignment of points to clusters that maximizes the sum of the scores associated to the instances and the final clusters they belong to, according to the allocation matrix, while satisfying the partition constraints and the user constraints.

This ILP model defines a Boolean variable matrix Z of size $N \times K$ such that $Z_{ik} = 1$ when the point i is assigned to the cluster k. Thus, if a point i is assigned to the cluster k then

Algorithm 2: AllocationMatrixComputation

Data: distance matrix $distMat$, number of clusters K, percentage per, number of instances N, partition P^* with K clusters
Result: M: an allocation matrix

begin

  LA = AnchorComputation($distMat, K, per, P^*$)
  D, R, M: matrices of dimension $N \times K$

  // computing the minimal distance from each point to each cluster (represented by the closest anchor)
  for i *from 1 to* N **do**
    for k *from 1 to* K **do**
      $D_{ik} = min_{a \in LA, P^*[a]=k}\, distMat[i, a]$

  //Normalization
  for i *from 1 to* N **do**
    for k *from 1 to* K **do**
      $R_{ik} = D_{ik} / \sum_{k' \in [1,K]} D_{ik'}$

  //Inversion
  for i *from 1 to* N **do**
    for k *from 1 to* K **do**
      $M_{ik} = (1 - R_{ik})/(K - 1)$

  Return M

$Z_{ik} = 1$ and $Z_{ik'} = 0$ for all the other $k' \neq k$. The partition constraints enforce that the solution is a partition:

1) $\forall k \in \{1, .., K\}, \sum_{i=1}^{N} Z_{ik} \geq 1$ (all the the clusters contain at least one element)
2) $\forall i \in \{1, .., N\}, \sum_{k=1}^{K} Z_{ik} = 1$ (all the elements belong to one and only one cluster)

User constraints are formulated on the variables Z. A must-link constraint on two points i, j is therefore represented by $Z_{ik} = Z_{jk}$ for all $k \in \{1, .., K\}$. A cannot-link constraint on i, j is represented by $Z_{ik} + Z_{jk} \leq 1$ for all $k \in \{1, .., K\}$. A triplet constraint (a, p, n) means that the point a is closer to p than to n, therefore if a and n are in the same cluster, p must be in that cluster also. This constraint on (a, p, n) is represented by $Z_{pk} \geq Z_{ak} + Z_{nk} - 1$ for all $k \in \{1, .., K\}$. This formulation yields $Z_{pk} = 1$ as soon as $Z_{ak} = 1$ and $Z_{nk} = 1$.

The problem is therefore formulated by:

$$\begin{array}{c} \arg\max_Z \sum_{i=1}^{N} \sum_{k=1}^{K} M_{ik} Z_{ik} \\ \text{s.t. partition constraints and user constraints} \end{array} \quad (1)$$

The result of the ILP model gives the best partition optimizing Equation (1) while satisfying all the constraints.

TABLE I: Characteristics of the data sets

Name	instances	dimensions	clusters
Halfmoon	200	2	2
ds2c2sc13	566	2	13
complex9	3088	9	2
iris	150	3	3
glass	277	9	7

IV. Experiments

We compare our method with four frameworks: when the base partitions are generated by K-Means (without constraints), when the base partitions are built with COP-kmeans [20] (thus integrating pairwise constraints), with a standard consensus partition building, by modifying the co-occurrence matrix for handling pairwise constraints. For all the experiments, the Euclidean distance is used.

In this section we show that:

- our method satisfies all the given constraints, whether it be pairwise or triplet constraints,
- our method does not reduce meaningfully the quality of the partition, and
- the ACCE method with a percentage of anchors of 25% gives the most similar results compared to the unconstrained consensus partition.

Our system has been developed in Python3 with the *sklearn* and *matplotlib* libraries. For generating base partitions with constraints, we use an existing COP-Kmeans code [21]. We compare our method with the Evidence Accumulation approach and a constrained version of the Evidence Accumulation [13] where if o_i and o_j are linked by an ML constraint, $S_{ij} = 1$ and if they are linked by a CL constraint $S_{ij} = 0$. If they are not linked by constraints then S_{ij} is not modified. The two methods will be respectively abbreviated by EAC and $ConstrEAC$ in our results. The used datasets are presented in Table I. `Halfmoon`, `ds2c2sc13` and `complex9` are synthetic datasets specially designed to contain several different cluster shapes.

A. Base partition generation

We generate a number $nBP = 50$ of base partitions with K-Means algorithm (or COP-Kmeans algorithm) with the number of clusters k selected randomly in the interval $\{2, min(50, \sqrt{n})\}$ where n is the number of elements in the dataset.

To control the randomness effect on our experiments, we create ten different base partition sets on which we perform the same type of experiments.

B. Constraints generation

The experiments are conducted with pairwise or triplet constraints.

To generate pairwise constraints, we randomly select two points in the dataset and add a ML constraint between them if they have the same ground-truth label, and a CL if not. We

TABLE II: ARI results with triplet constraints

Dataset	K-means	EAC	ILP25	ILP10	ILP5
halfmoon	0.256	0.998	0.995	0.995	0.907
ds2c2sc13	0.551	0.591	0.587	0.578	0.571
complex9	0.414	0.776	0.775	0.774	0.760
iris	0.730	0.566	0.560	0.570	0.570
glass	0.260	0.268	0.259	0.237	0.251

TABLE III: Percentage of satisfied triplet constraints

Dataset	K-means	EAC	ILP25	ILP10	ILP5
halfmoon	90.00	99.96	100.0	100.0	100.0
ds2c2sc13	99.76	99.82	100.0	100.0	100.0
complex9	99.02	99.34	100.0	100.0	100.0
iris	97.80	98.60	100.0	100.0	100.0
glass	93.84	95.30	100.0	100.0	100.0

create 5 constraint sets with 50 constraints each, and each of them will be used with the 10 base partition sets.

Similarly, to generate triplet constraints we use the ground-truth labels and the Euclidean distance matrix. We start by selecting at random 3 different instances a, p and n. If they are all in the same cluster or all in different clusters, then we create a constraint (a, p, n) if a is closer to p than to n or (a, n, p) otherwise. If a and p are in the same cluster but not n we add the triplet constraint (a, p, n), and if a and n are in the same cluster but not p then we add the triplet constraint (a, n, p).

C. Number of anchors

For the number of anchors, we compare the results obtained with 3 different percentages: 25%, 10% and 5%. In the results they will be referred to as ILP-Km25, ILP-Km10 and ILP-Km5 when using the K-Means algorithm to generate the base partitions, and ILP-COP25, ILP-COP10 and ILP-COP5 with COP-Kmeans.

D. Evaluation criterion

The results are evaluated based on two criteria: the constraint satisfaction and the quality of the final partition. With respect to partition quality, we use the Adjusted Rand Index (ARI). Given a partition P_i generated with a clustering algorithm for a dataset and a partition P^T representing the true partition/ground-truth, this measure determines the similarity between P_i and P^T by analyzing the pairwise co-assignment of points between the two partitions. The upper bound of ARI is 1, indicating a perfect agreement between the partitions, and values near 0 or negative correspond to cluster agreement found by chance [22].

The equation computing the ARI is the following:

$$ARI(P_i, P^T) = \frac{2(ab - cd)}{(a+d)(d+b) + (a+c)(c+b)} \quad (2)$$

with

- a: number of pairs of instances clustered together in P_i and in P^T,
- b: number of pairs of instances clustered together in P_i but not in P^T,
- c: number of pairs of instances clustered together in P^T but not in P_i,
- d: number of pairs of instances clustered together neither in P_i nor in P^T.

E. Results

We present the results in Figs. 3 and 4. Let us recall that we generate 10 sets of base partitions and 5 sets of constraints, leading to 50 results. The boxplots give a summary in terms of quantiles of the 50 results. The figures in cyan are the results obtained from non-ensemble methods K-Means and COP-Kmeans. The blue ones are from ensemble methods (with or without constraints): Base partitions (indicated by BP.KM), the evidence accumulation (EAC), the Constrained Evidence Accumulation (Constr EAC) and Evidence Accumulation when the base partitions are generated with COP-Kmeans (EAC COP). The red ones give the results of our method: ILP-Km25, ILP-Km10 and ILP-Km5 (base partitions generated by K-means), ILP-COP25, ILP-COP10 and ILP-COP5 (base partitions generated using COP-Kmeans); the number in each case represents the percentage of the points mapped into anchors. In all ensemble methods, the consensus partition, at the basis of our work, is generated by the Evidence Accumulation algorithm.

From these results, we can observe that our ACCE method with a percentage of anchors of 25% allows to satisfy the constraints without deteriorating significantly the quality of the original EAC partition.

With 5% of anchors, the results seem to be more dependent on the shape of the dataset being sometimes slightly better or worse in terms of ARI than the initial EAC partition.

Our method was experimented with the base partitions generated either with K-Mmeans or COP-Kmeans. We can notice that using COP-Kmeans algorithm the results usually have a larger variance compared to those obtained using K-Means.

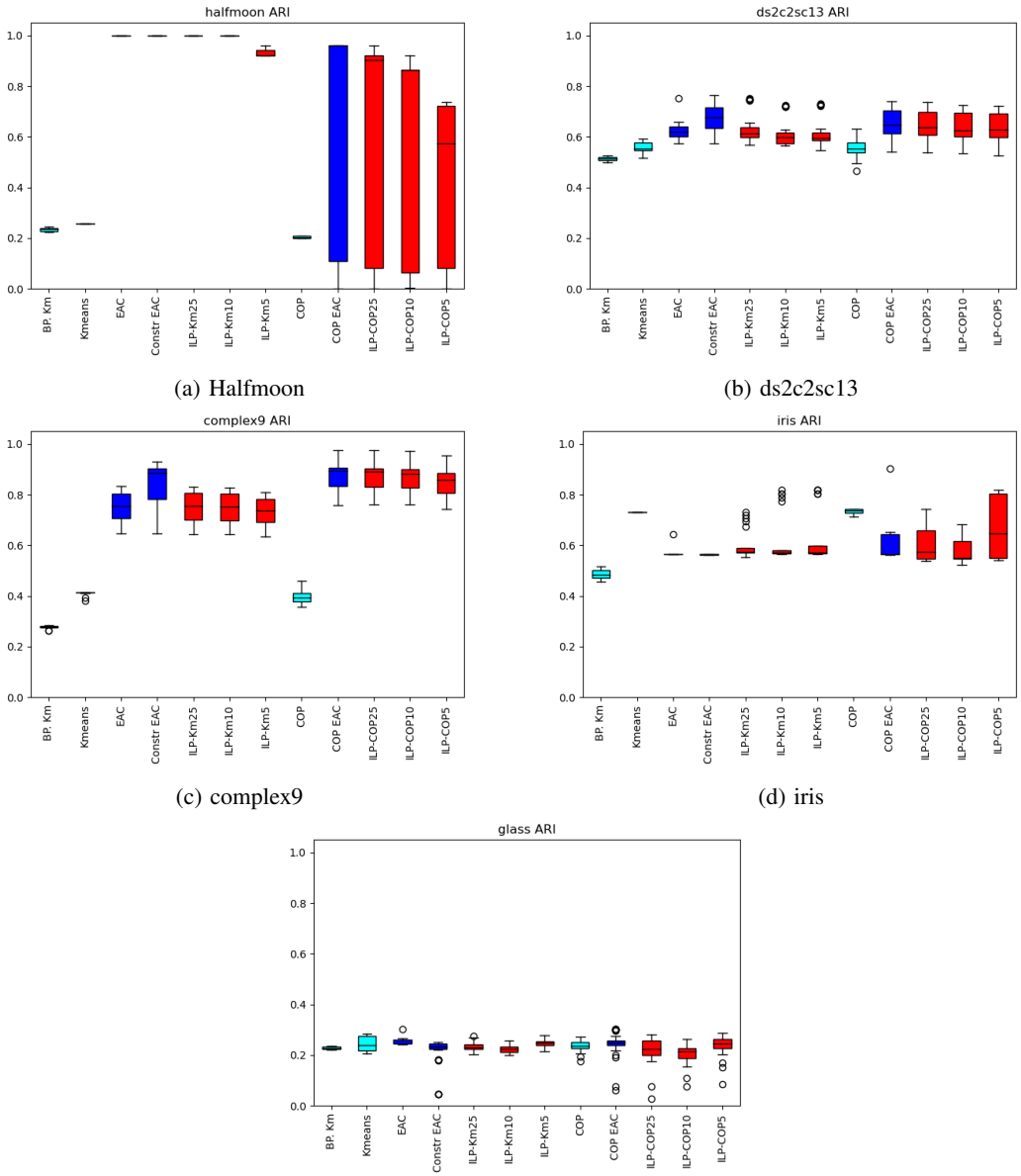
Concerning the triplet constraints, from Tab. II and Tab. III we see that ACCE ensures the satisfaction of all the constraints, in contrast to the unconstrained methods we tested, which cannot handle this kind of constraints.

V. Conclusion and Future Work

A. Conclusion

In this work, we present a novel Constrained Clustering Ensemble approach integrating for the first time the constraints after the generation of the consensus partition and show that it can take into account pairwise and triplet constraints. Such a framework could also be adapted to other types of constraints.

Experiments show that this method allows to satisfy the constraints without deteriorating significantly the quality of the unconstrained consensus partition. We draw the attention



(a) Halfmoon

(b) ds2c2sc13

(c) complex9

(d) iris

(e) glass

Fig. 3: Adjusted Rank Index for different datasets with pairwise constraints

on the fact that in this work the constraints were generated at random, and may not be sufficiently informative to improve the results in terms of ARI.

Unlike other constrained ensemble clustering methods, our approach ensures the satisfaction of all the given constraints.

B. Future Work

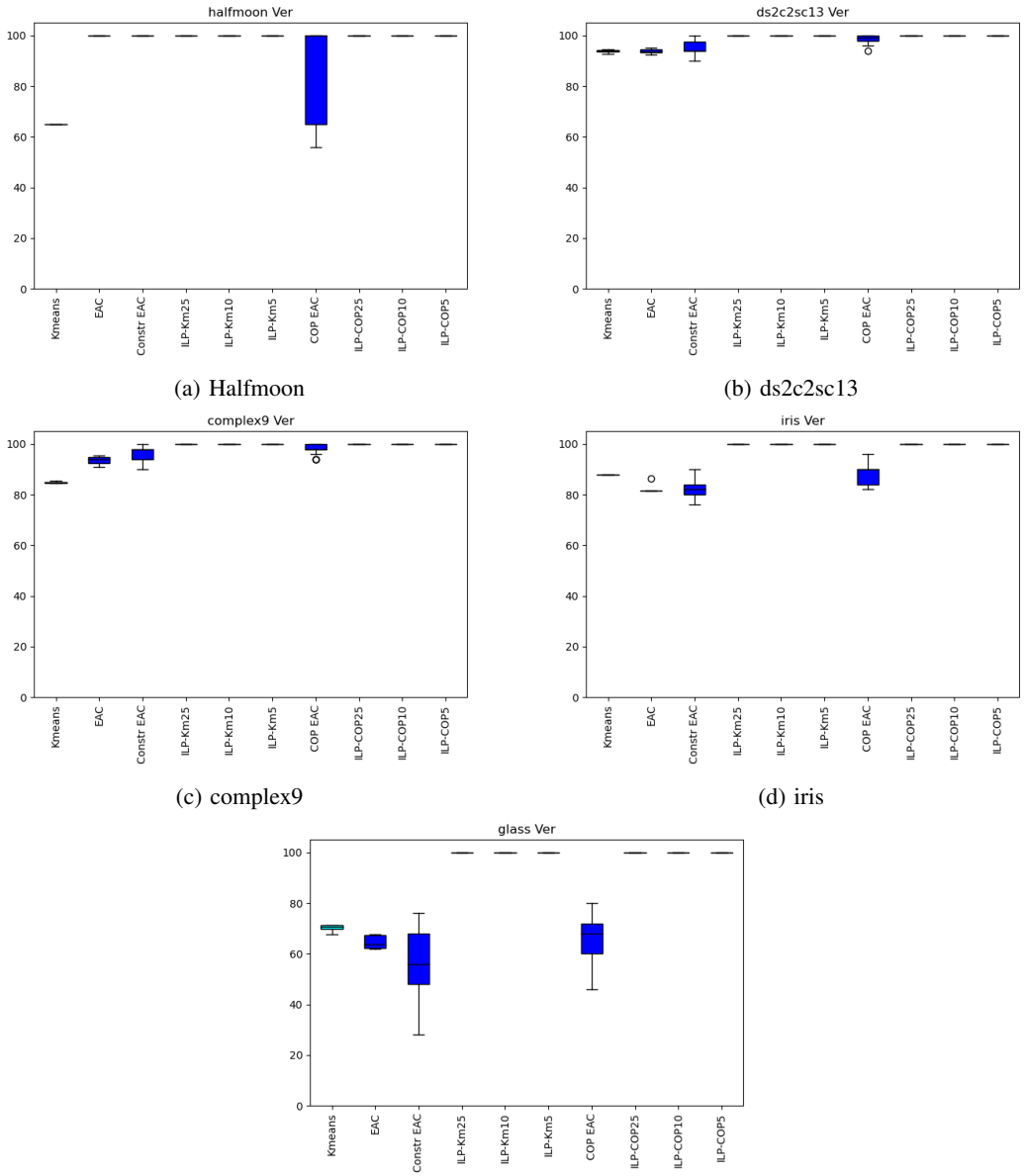
As future work, we intend to explore other ways to compute the anchors so that it would be adaptive to the size and density of each cluster. We could also compare our performance obtained by using a single link algorithm versus other types of hierarchical clustering.

Concerning the constraints, we can integrate other types of constraints in the ILP model. Considering multiple constraint types at the same time can easily be achieved in our model. One of the main difficulties is to find relevant (informative) constraints that have a true impact on the clustering. Another direction would be to study the generation of informative constraints, as done for instance in active learning for pairwise constraints.

ACKNOWLEDGMENTS

This work was funded by the ANR project InvolvD (Interactive constraint elicitation for unsupervised and semi-supervised data mining) (ANR-20-CE23-0023).

The authors would also like to thank Nghiem Nguyen Viet Dung for his help on the utilization of his ILP model [8].



(a) Halfmoon

(b) ds2c2sc13

(c) complex9

(d) iris

(e) glass

Fig. 4: Percentage of satisfied constraints for different datasets with pairwise constraints

NOTES

The code associated to this article is available at https://github.com/MathieuGuilbert/ACCE .

REFERENCES

[1] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[2] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[3] A. Strehl and J. Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617, 2002.

[4] T.-B.-H. Dao, C. Vrain, K.-C. Duong, and I. Davidson. A framework for actionable clustering using constraint programming. In *Proceedings of the Twenty-second European Conference on Artificial Intelligence*, pages 453–461, 2016.

[5] B. Babaki, T. Guns, and S. Nijssen. Constrained clustering using column generation. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 438–454. Springer, 2014.

[6] I. Davidson, SS. Ravi, and L. Shamis. A sat-based framework for efficient constrained clustering. In *Proceedings of the 2010 SIAM international conference on data mining*, pages 94–105. SIAM, 2010.

[7] K.-C. Duong T.-B.-H. Dao and C. Vrain. Constrained clustering by constraint programming. *Artificial Intelligence*, 244:70–94, 2017.

[8] N.-V.-D. Nghiem, C. Vrain, T.-B.-H. Dao, and I. Davidson. Constrained clustering via post-processing. In *International Conference on Discovery Science*, pages 53–67. Springer, 2020.

[9] A. LN Fred and A. K. Jain. Combining multiple clusterings using evidence accumulation. *IEEE transactions on pattern analysis and*

machine intelligence, 27(6):835–850, 2005.

[10] S. Basu, I. Davidson, and K. Wagstaff. *Constrained clustering: Advances in algorithms, theory, and applications*. CRC Press, 2008.

[11] Yue Qin, Shifei Ding, Lijuan Wang, and Yanru Wang. Research progress on semi-supervised clustering. *Cognitive Computation*, 11(5):599–612, 2019.

[12] D. Chen, Y. Yang, H. Wang, and A. Mahmood. Convergence analysis of semi-supervised clustering ensemble. In *2013 IEEE Third International Conference on Information Science and Technology (ICIST)*, pages 783–788. IEEE, 2013.

[13] L. Sun J. Yang and Q. Wu. Constraint projections for semi-supervised spectral clustering ensemble. *Concurrency and Computation: Practice and Experience*, 31(20):e5359, 2019.

[14] Z. Yu, H. Chen, J. You, H.-S. Wong, J. Liu, L. Li, and G. Han. Double selection based semi-supervised clustering ensemble for tumor clustering from gene expression profiles. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 11(4):727–740, 2014.

[15] Z. Yu, Z. Kuang, J. Liu, H. Chen, J. Zhang, J. You, H.-S. Wong, and G. Han. Adaptive ensembling of semi-supervised clustering solutions. *IEEE Transactions on Knowledge and Data Engineering*, 29(8):1577–1590, 2017.

[16] Z. Yu, P. Luo, J. Liu, H.-S. Wong, J. You, G. Han, and J. Zhang. Semi-supervised ensemble clustering based on selected constraint projection. *IEEE Transactions on Knowledge and Data Engineering*, 30(12):2394–2407, 2018.

[17] S. Wei, Z. Li, and C. Zhang. Combined constraint-based with metric-based in semi-supervised clustering ensemble. *International Journal of Machine Learning and Cybernetics*, 9(7):1085–1100, 2018.

[18] F. Yang, T. Li, Q. Zhou, and H. Xiao. Cluster ensemble selection with constraints. *Neurocomputing*, 235:59–70, 2017.

[19] Y. Yang, H. Wang, C. Lin, and J. Zhang. Semi-supervised clustering ensemble based on multi-ant colonies algorithm. In *International Conference on Rough Sets and Knowledge Technology*, pages 302–309. Springer, 2012.

[20] Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schrödl, et al. Constrained k-means clustering with background knowledge. In *Icml*, volume 1, pages 577–584, 2001.

[21] B. Babaki. Cop-kmeans version 1.5, July 2017.

[22] L. Hubert and P. Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.