Pattern Recognition 45 (2012) 1749–1758

Contents lists available at SciVerse ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Improving constrained clustering with active query selection

Viet-Vu Vu [a,*], Nicolas Labroche [a], Bernadette Bouchon-Meunier [b]

[a] UPMC Univ Paris 06, UMR 7606, LIP6, F-75005 Paris, France
[b] CNRS, UMR 7606, LIP6, F-75005 Paris, France

ARTICLE INFO

Article history:
Received 14 February 2011
Received in revised form
14 September 2011
Accepted 5 October 2011
Available online 9 November 2011

Keywords:
Active semi-supervised clustering
Pairwise constraints
k-Nearest neighbors graph

ABSTRACT

In this article, we address the problem of automatic constraint selection to improve the performance of constraint-based clustering algorithms. To this aim we propose a novel active learning algorithm that relies on a k-nearest neighbors graph and a new constraint utility function to generate queries to the human expert. This mechanism is paired with propagation and refinement processes that limit the number of constraint candidates and introduce a minimal diversity in the proposed constraints. Existing constraint selection heuristics are based on a random selection or on a min–max criterion and thus are either inefficient or more adapted to spherical clusters. Contrary to these approaches, our method is designed to be beneficial for all constraint-based clustering algorithms. Comparative experiments conducted on real datasets and with two distinct representative constraint-based clustering algorithms show that our approach significantly improves clustering quality while minimizing the number of human expert solicitations.

1. Introduction

In recent years, clustering with constraints (also known as clustering with side information) has become a topic of significant interest for many researchers. These methods allow to take into account a user's knowledge expressed as a set of constraints to improve the clustering results. There exist several families of constraints but the most used are: must-link (ML) and cannot-link (CL) constraints [1]. On the one hand, ML constraints indicate that two points of the dataset have to be grouped in the same cluster. On the other hand, CL constraints impose that the points belong to different clusters. Constraints are also used in other domains such as constrained classification [2,3], and feature selection [4].

We can divide previous works on clustering with constraints into two main families: either (1) *distance-based methods*: the constraints help the algorithm to learn a metric/objective function [5–12] or (2) *constraint-based method*: the constraints are used as *hints* to guide the algorithm to a useful solution [1,13–18].

Following [19], given a constraint set, the *distance-based methods* are first trained to "satisfy" the constraints so that, after training, data objects associated by a must-link constraint should be close and data objects linked by a cannot-link constraint should be well separated in the learning space. Some distance measures have been used for distance-based constrained clustering: string-edit distance trained using EM, Jensen–Shannon divergence trained using gradient descent, Euclidean distance modified by shortest-path algorithm, and Mahalanobis distance trained using convex optimization. Some recent techniques include learning a distance metric transformation that is globally linear but locally non-linear, and learning a margin-based clustering distortion measure using boosting.

In *constraint-based approaches*, two families of methods can be found: on the one hand, algorithms with a strict enforcement, which find the best feasible clustering respecting all the constraints, and, on the other hand, algorithms with partial enforcement, which find the best clustering while maximally respecting the constraints. To this aim, several techniques have been proposed so far in the literature: modifying the clustering objective function so that it includes a term of constraint satisfiability, enforcing all constraints to be satisfied during the assignment step in the clustering process, or initializing clusters and inferring clustering constraints based on neighborhoods derived from labeled example [19].

The motivation of our work focuses on two open questions that follow:

1. *How can we determine the utility of a given constraint set, prior to clustering* [20]? The need for a constraint utility measure has become imperative with the recent observation that some poorly defined constraint sets can decrease clustering performance [20–22]. In this article, we define a set of desirable properties for such a utility measure and we propose a first implementation based on these properties. Our measure evaluates the ability of a constraint

* Correspondence to: LIP6 - Université Pierre et Marie Curie - Paris 6 UMR CNRS 7606, 4 place Jussieu, case 169, 75252 Paris cedex 05, France.
Tel.: +33 1 44 27 88 87; fax: +33 1 44 27 70 00.
E-mail addresses: viet-vu.vu@lip6.fr (V.-V. Vu), nicolas.labroche@lip6.fr (N. Labroche), bernadette.bouchon-meunier@lip6.fr (B. Bouchon-Meunier).

0031-3203/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2011.10.016

to help the clustering algorithm to distinguish the points in the perturbation regions, e.g. sparse regions or transition regions where clustering algorithms traditionally perform poorly. Finally, we use this measure to develop an active constraint selection algorithm.

2. *How can we minimize the effort required from the user, by only soliciting her (him) for the most useful constraints* [20,23]? Many researches have been conducted on the problem of clustering with constraints [7,10,13–15,17,24] but most of the time the user is supposed to provide the algorithm with good constraints in a passive manner. One alternative is to let the user actively choose the constraints. However, as some poorly chosen constraints can lead to a bad convergence of the algorithms [21,22] and as there is possibly $(n\times(n-1))/2$ ML or CL constraints in a dataset with n points, the choice of the constraints appears to be a crucial problem. Some works are proposed on this topic but they only focus on K-Means clustering [22,25].

This paper extends our preliminary researches [26,27] and presents a new active constraint selection algorithm to collect a constraint set that can improve the performance of any constraint-based clustering algorithm such as Constrained K-Means [1,7,14], Constrained-DBSCAN [15], Constrained Fuzzy C-Means [16], Constrained Hierarchical Clustering [13], Constrained Spectral Clustering [28] or Constrained Leader Ant Clustering [17]. Our method relies on a k-nearest neighbors graph to estimate sparse regions of the data where cluster membership is most uncertain and where constraints should be most beneficial.

The rest of the paper is organized as follows: Section 2 discusses the related work. Section 3 presents our new framework for active constraint selection, while Section 4 describes the experiments that have been conducted and the comparative results on benchmark datasets. Finally, Section 5 concludes and discusses future research.

2. Related work

There are few works on active constraint selection for clustering. In contrast, active learning for classification has a long history, in which different strategies to select the queries have been used [19]: reduction of the version space size, reduction of uncertainty in predicted label, maximization of the margin on training data, finding high variance data instances by density-weighted pool-based sampling, etc.

In [25], an algorithm for active constraint selection for K-Means using farthest-first strategy is proposed. This algorithm is referred to as the Farthest First Query Selection (FFQS) [25]. The FFQS algorithm has two phases: *Explore* and *Consolidate*.

The *Explore* phase defines a set of CL constraints under the strong hypothesis that, at the end of this phase, at least one point has been drawn in each cluster. To achieve this goal, the method needs to know the number of clusters in the partition. The set of CL constraints is called the *skeleton* of the clusters. At each iteration, the farthest point from existing *skeleton* is chosen as a candidate for a CL query.

The second phase *Consolidate* randomly picks a point not in the *skeleton* and queries it against each point in the skeleton, until the user decides to stop labeling the constraints.

In [22], an improved version of FFQS using a min–max approach is proposed. We refer this approach as MMFFQS method. The *Explore* phase is similar as the FFQS approach and relies on the same strong hypothesis. In the *Consolidate* phase, instead of randomly selecting the point, the idea is to select the data point whose largest similarity to the *skeleton* is the smallest. By this way, data points with largest uncertainty in cluster membership are chosen first to express the user queries.

Both previous methods do not work well in the case of a dataset with a large number of clusters or unbalanced datasets with small clusters. However, as this method improves the simple FFQS approach [25], it is used as a baseline method in our experiments.

Finally, in [29], Xu et al. propose active constraint selection for spectral clustering. The key idea of this work is to use the theory of spectral decomposition to identify data items that are likely to be located on the boundaries of clusters. However, the authors only focus on two-clusters problem.

3. Active constraint selection framework

In order to collect a suitable constraint set, our algorithm first builds a set of candidate constraints from a k-Nearest Neighbors Graph (k-NNG). Second, our approach uses of a new constraint set utility measure to rank the candidate constraints according to their ability to separate clusters.

3.1. The k-nearest neighbors graph

We define the k-NNG as a weighted undirected graph, in which each vertex represents a data point, and possesses at most k edges to its k-nearest neighbors. An edge is created between a pair of vertices, u and v, if and only if the points associated to vertices u and v have each other in their k-nearest neighbors set. The weight $\omega(u,v)$ of the edge between the vertices u and v is defined as the number of common nearest neighbors the two points associated to u and v share, as shown in Eq. (1) [30]

$$\omega(u,v) = |NN(u)\cap NN(v)| \tag{1}$$

where $NN(\cdot)$ denotes the set of k-nearest neighbors of the associated point. The important property of this similarity measure is its own built-in automatic scaling, which makes it adapted to treat datasets with distinct cluster densities.

3.2. A new measure of constraint utility

In this section, we propose a first definition of a utility measure for selecting constraints. This measure relies on the hypothesis that a constraint is valuable if it can help clustering algorithms to better separate the points in the sparse regions or transition regions between clusters, where the cluster membership is most uncertain. Thus, the idea is to select couples of points so that the associated constraints can help reducing the clustering ambiguity. To this aim we express the following hypothesis under which our definition relies.

Hypothesis 1. It is possible to define a constraint utility measure between two points from the density of the regions where the points belong to and from an expression of the distance between the points.

The k-NNG representation already proposes a distance relation between two points with the weight ω that indicates the number of common neighbors between the points (see Eq. (1)). In the k-NNG defined above, edges with small ω values are isolated from the other points and thus belongs to sparse or transition regions between clusters.

Our proposal is then to use the Local Density Score (*LDS* for short) introduced by [31], as an indicator of the density of the region of a vertex u. Given a k-NNG, the *LDS* of a vertex $u\in k$-NNG is defined by Eq. (2)

$$LDS(u) = \frac{\sum_{q\in NN(u)}\omega(u,q)}{k} \tag{2}$$

The *LDS* score of a point in $[0,k-1]$ is the average distance to its k-nearest neighbors. *LDS* is defined in such a way that a high value indicates a strong association between the point u and its neighbors, i.e. u belongs to a dense region. In contrast, a low value of *LDS* means that u belongs to a sparse region or transition region between clusters.

Both previous indicators ω and *LDS* have small values when the points are in sparse regions, and tend to have large values when the points fall into dense regions of the datasets. In order to better appreciate the relation that may exist between these indicators, Fig. 1 plots the values of ω versus the values of *LDS* for the dataset Soybean from the UCI Machine Learning Repository [32]. It can be observed from this figure that when the values of *LDS* are high for both points, the value of ω is also high, meaning that when both points are in dense regions their connectivity is likely to be high. Similarly, if the values of *LDS* are low then points are in sparse regions and the weight ω is likely to have a small value. This illustration shows that *LDS* and ω values seem to be correlated to some extent, but it can be also observed from Fig. 1 that some couples of points can have a high ω value while one of the points in the couple exhibits a small *LDS* value. This observation leads us to keep both indicators *LDS* and ω to specify our constraint utility function as described hereafter.

From the previous observations, we define, in our approach, a constraint utility function as a combination of monotonic decreasing functions of ω and *LDS*. As an example, we propose the first definition of a novel utility measure of a constraint between two points u and v as an Ability to Separate between Clusters (*ASC*) as shown in Eq. (3)

$$ASC(u,v) = k - \omega(u,v) + \frac{1}{1+\min\{LDS(u), LDS(v)\}} \tag{3}$$

As stated before, this constraint utility function depends on two indicators: the weight $\omega(u,v)$ and the density *LDS* of the region each point belongs to. We define the density of a constraint between two points u and v as the minimum between $LDS(u)$ and $LDS(v)$ which means that if, at least, one of the two points is in a sparse region, then the score of the constraint is likely to be large. Of course, with this formulation, there is no difference between a constraint where both points are in sparse regions and constraints where only one point has a low Local Density Score. It is also important to notice that this expression gives more discrimination power to ω than *LDS* since the former is defined in $[0,k]$ and the latter in $[0,1]$. The reason is that we first evaluate the constraints on the basis of the ω score. Then, when several constraints have similar ω scores, the *LDS* score is used to differentiate the candidates. Other choices could be made for the formalization of the *ASC* score using various aggregation measures to balance the contribution of ω and *LDS*. As an illustration, Section 4.4 presents comparative results with an alternative ASC score that relies on a product rather than a sum between the ω term and the *LDS* term. This alternative score gives more weight to the *LDS* term and proposes an elegant weighting scheme. However, as its results are not as good as those of the ASC score proposed in Eq. (3) and because of space limitation, it is not detailed further in the rest of the paper and only used to illustrate the benefit of our proposal with the AHCC clustering algorithm [13] in Section 4.4.

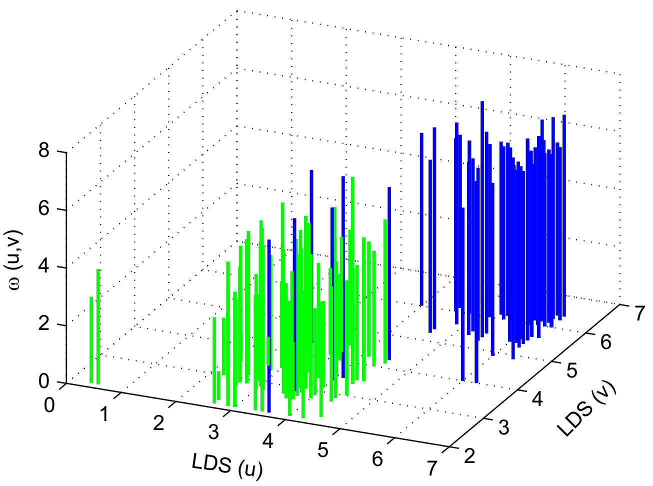


Fig. 1. $LDS(u)$, $LDS(v)$ and $\omega(u,v)$ for Soybean dataset. The blue lines have the $\omega(u,v) > 5$ and the green lines have the $\omega(u,v) \le 5$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Experiments reported in Section 4 give more details on how the indicators ω and *LDS* contribute to the clustering performance in the *ASC* expression, and show that a constraint with high *ASC* score is more likely to provide performance gains than a randomly chosen one.

Finally, it can be noticed that the *ASC* score, as it is defined locally between two points, cannot bear information about the novelty or diversity of the candidate constraint respective to the constraints that may have been already provided or computed by our method. To this aim, two mechanisms (namely the *Propagation* and the *Refinement*) have been proposed in our framework to avoid considering constraints that are in the same transition regions. These mechanisms are described in Section 3.4.

3.3. Identification of candidate constraints

Following the principle of active learning [33], we choose the data points to express our queries in the sparse regions or transition regions, i.e. where the ω values are the smallest. In order to limit the number of constraint candidates in the set of candidates C (see Eq. (4)), we filter the edges in the k-NNG according to a threshold θ as part of the queries selection process. The set of candidates C is defined as follows:

$$C = \{(u,v) \,|\, weight(u,v) < \theta\} \tag{4}$$

Two approaches are proposed to generate the user queries: either the candidates are randomly chosen from the set C or the candidates are ordered according to the *ASC* score so that the candidates that maximizes the *ASC* score are considered first in the queries selection process.

3.4. Active constraint selection algorithm

As stated before, our approach builds the set of candidate constraints as the set of all edges in the k-NNG whose weights are under the threshold value θ. The active constraint selection is expressed as a loop until the entire candidate set is examined or the user stops. At each iteration, the algorithm picks a candidate constraint between points u and v from the set of remaining candidates following the conditions in Section 3.3 and asks the user about the nature of the relation: Must-Link (ML), Cannot-Link (CL) constraint or "I don't know". If the answer is ML or CL, it defines a new constraint of that type between points u and v named $Label(u,v)$ and stores it in the set of final constraints (see Algorithm 1). If the answer is "I don't know", the constraint is simply discarded from the set of candidate constraints. In our experiments where every data labels are known, the answer can only be ML or CL (see Section 4).

Contrary to other active learning methods for constraint-based clustering, our approach proposes *propagation* and *refinement* mechanisms. These mechanisms aim, on the one hand, to discover new constraints from the set of candidates and the set of existing constraints and, on the other hand, to reduce the number of constraint candidates.

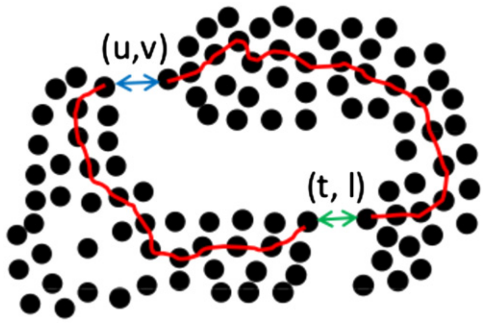


Fig. 2. Illustration of the propagation mechanism: *Label*(t,l) is propagated as *Label*(u,v) because of the presence of strong paths $SP(u,t,\theta)$ and $SP(v,l,\theta)$ between them.

3.4.1. *Propagation mechanism*

Given a set of candidate constraints *C* and a set of constraints *Y*, the *Propagation* procedure discovers new constraints in *C* from the information stored in *Y*. To this aim, we first define the notion of a *strong path*.

Definition 1. Given a *k*-nearest neighbors graph (*k*-NNG) for a dataset *X*, a threshold θ, and a set of constraints *Y* (*Y* can be empty), a path from vertex *u* to vertex *v* is defined as a strong path $SP(u,v,\theta)$ if there exists a sequence of vertices $(z_1,z_2,\ldots,z_t)$ such that $u=z_1$, $v=z_t$ and $\forall i=1\ldots t-1$: $\omega(z_i,z_{i+1})\geq\theta$ or (z_i,z_{i+1}) is a must-link constraint in *Y*.

Two main rules are then applied to propagate new constraints in *Y* from candidates *C*.

First, given a constraint (u,v) in *Y* and a candidate (t,l) in *C*, if there exists strong paths: $SP(u,t,\theta)$ and $SP(v,l,\theta)$ or $SP(u,l,\theta)$ and $SP(v,t,\theta)$ then constraint (t,l) is added to *Y* and $Label(t,l)=Label(u,v)$ (see Fig. 2). This rule is based on the hypothesis that if there exists a constraint between two points from two clusters, then any couples of points from these two clusters should exhibit the same constraint. As the propagation process is an in line mechanism that operates before the final clusters are known, the cluster membership is replaced by an estimation according to the Strong Path formalism. It can be noticed that the latter is somewhat similar to the notion of *density reachability* found in the DBSCAN algorithm [34].

Second, given two constraints (u,v) and (v,w) in *Y*, we have, according to [22]: (*i*) $ML(u,v)\wedge ML(v,w)\Rightarrow ML(u,w)$ and (*ii*) $ML(u,v)\wedge CL(v,w)\Rightarrow CL(u,w)$. These new generated constraints are then added to *Y*.

3.4.2. *Refinement procedure*

The *Refinement* procedure reduces the size of the candidate set by removing all constraints between two points that are likely to be in the same cluster. To that aim, the *Refinement* removes constraint candidates that are linked by a *strong path*. The objective is to identify points that are indirectly connected through a dense region, similarly to the propagation of labeled data in semi-supervised classification [35] or in the algorithm DBSCAN [34]. This procedure is crucial for the performance of our approach since it decreases the size of the constraint candidates set.

The main steps of our algorithm are summed up in Algorithm 1 that gives a general overview of our approach.

Algorithm 1. Active constraint selection

```
Require: Dataset $X=\{x_i\}_{i=1}^n$, k and threshold $\theta$
Ensure: Set of collected constraints Y
1:  $Y=\emptyset$
2:  Construct a k-NNG of X
3:  $C=\{(u,v)|weight(u,v)<\theta\}$
4:  Refinement$(C,\theta)$
5:  while (UserStop=False) and $(C\neq\emptyset)$ do
6:    Pick a $(u,v)\in C$ following Section 3.3
7:    Query the user about the Label of $(u,v)$?
8:    if Label is ML or CL then
9:      $Y=Y\cup\{Label(u,v)\}$
10:     Propagation$(C,Y,\theta)$
11:     Refinement$(C,\theta)$
12:   else
13:     Save $(u,v)$ so that the query is not asked again later
14:   end if
15: end while
```

The complexity of our algorithm depends on the complexity to build the *k*-NNG and on the complexity of the refinement procedure. The complexity of building the *k*-NNG is linear with the time for a *k*-nearest neighbors query. Following Breunig et al. [36], for *k*-nearest neighbors queries, we have a choice among different methods. For low-dimensional data, we can use a grid based approach which answers *k*-nearest neighbors queries in constant time and leads to an overall complexity of $O(n)$ to build the *k*-NNG. For medium to medium high-dimensional data, we can use an index, which leads to an overall complexity of $O(\log n)$. Finally, for extremely high-dimensional data, we need to use a sequential scan or some variant of it, e.g. the VA-file [37], with a global complexity of $O(n^2)$ to build the *k*-NNG. The complexity of the *Refinement* procedure is $O(n*k)$ (scan of all the edges of *k*-NNG).

So, the complexity of our algorithm is $O(n*k)$, $O(n*\log n)$ or $O(n^2)$ depending on the dimension of the data that can be low, medium or extremely high. Section 4.6 discusses the computational efficiency of our new constraint selection framework based on a *k*-NNG when compared to the Min–Max FFQS method [22].

4. Experiments

4.1. *Experimental protocol*

Our new active query selection framework based on a *k*-NNG is evaluated in two ways: first it is compared to other constraint selection heuristics like the MMFFQS method [22] and second, it is used in conjunction with two different constraint-based clustering algorithms to show the adaptability of the proposed approach.

4.1.1. *Constraints selection heuristics*

In our study we consider the following heuristics to select the constraints:

1. "*Proposed-ASC*" corresponds to our active constraint selection algorithm in which the candidates issued from the *k*-NNG are selected according to the *ASC* score (see Section 3.3). This method is deterministic, so the tests can be performed only once. Similarly "Proposed-ASC prod" corresponds to the alternative product based ASC score mentioned in Section 3.2.
2. "*Proposed Random*" corresponds to our active constraint selection algorithm in which the candidates issued from the *k*-NNG are randomly selected. As this approach is non-deterministic, the associated results are averaged over 50 runs for each dataset and each clustering algorithm.
3. "*MMFFQS*" refers to the Min–Max FFQS heuristic [22]. This method is deterministic, so we only need to perform it once.
4. "*Random*" corresponds to a completely random selection of the constraints that are proposed to the human expert. This method generates a set of ML and CL constraints based on the comparison of the labels of randomly chosen pairs of

objects. If both labels are in the same cluster, a ML constraint is generated, and else, a CL constraint is generated. As this approach is non-deterministic, the results are averaged over 50 runs for each dataset and each algorithm.

4.1.2. *Algorithms*

In our experiments we report the results obtained with two different constraint-based clustering algorithms: the Agglomerative Hierarchical Clustering with Constraints [13] (AHCC hereafter) and the constrained K-Means MPC-KMeans [7]. They have been chosen because they are representative of the most popular clustering algorithms in practice according to a study reported by Jain et al. [38]. As AHCC returns a dendrogram, its performances are evaluated on the basis of the partition from the dendogram that maximizes the clustering evaluation criterion (see Section 4.2) for each run.

4.1.3. *Datasets*

We use six well-known real datasets from the Machine Learning Repository [32] (each with the following number of objects, attributes and clusters): Iris (150/4/3), Glass (214/9/6), Breast (569/30/2), Soybean (47/34/4), Protein (116/20/6) and Image Segmentation (2100/19/7) to evaluate our algorithm. These datasets have been chosen because they facilitate the reproducibility of the experiments and because some of them have already been used in constraint-based clustering articles.

4.2. *Evaluation method*

As all our benchmark datasets contain a class label, we use the Rand Index (*RI* hereafter) [39] in our experiments to evaluate the agreement between the theoretical partition of each dataset and the output partition of the evaluated algorithms. *RI* takes values between 0 and 1; $RI=1$ when the result is the same as the ground-truth. The larger the *RI*, the better the result.

4.3. *Parameters setting*

Our active selection framework uses two parameters: the number of nearest neighbors k and the threshold θ. To evaluate the influence of these parameters on our new constraint selection heuristic, tests have been conducted with the AHCC algorithm paired with the "Proposed ASC" selection heuristic (see Section 4.1.1). As illustrated in Fig. 3-Left, the value of k cannot be generalized for all datasets, because it depends on the structure and the size of the datasets. For example, some datasets are very sensitive to the value of k because of overlapping clusters (see Iris dataset). Fig. 3-Right presents the RI measure obtained for values of θ ranging theoretically from 1 to $k-1$ (k is set experimentally for each dataset in this case). However, some values of θ cannot be considered in our experiments: lower values do not allow enough constraint candidates (see Eq. (4)) to perform our new constraint selection heuristic and higher values makes the selection heuristic intractable since there are too many constraint candidates and since the Refinement process cannot be applied efficiently because only few strong pathes may be defined. The experiments reported in Fig. 3-Right show that, as for the k parameter, it is difficult to predict the best value for the parameter θ: for some datasets the results only show little variations (Soybean, Glass, Protein, Image Segmentation) while some others seem to be more sensitive (Iris, Breast). As a conclusion, we propose to set the θ parameter in $[\lceil (k/2)-2\rceil, \lfloor (k/2)+2\rfloor]$ which gives the best results on our benchmark datasets and, more generally, should allow the selection heuristics to be applied efficiently.

4.3.1. *Detailed analysis of the ASC constraint utility measure*

The experiments reported in this section aim at evaluating the individual contribution of the indicators ω and *LDS* to the constraint utility function *ASC* proposed in Section 3.2 in terms of clustering performance. To this aim, tests have been conducted on our benchmark datasets with the AHCC algorithm paired with either the "Proposed ASC" constraint selection heuristic or with two selection heuristics derived from "Proposed ASC" but that only rely on one indicator as follows:

1. the first one depends on ω

$$OmegaOnly(u,v) = k-\omega(u,v)$$

2. the second one depends on *LDS*

$$LDSOnly(u,v) = \frac{1}{1+\min\{LDS(u),LDS(v)\}}$$

As illustrated in Fig. 1, ω and *LDS* seem to partially hold the same information relatively to our constraint utility function. However, it can be observed from Fig. 4 that the contribution of ω is more significant in our constraint selection process when applied to the clustering problem than the contribution of *LDS*. For all of our benchmark datasets, the performances of *LDSOnly* measure are inferior to those observed for *OmegaOnly*. The latter are sometimes equivalent to those of the *ASC* measure (for *Iris* dataset for example) and also locally slightly better for a given number of queries and a given dataset (for example 40 queries with *Glass* dataset). In these cases, it is possible that *ASC* value may be lowered by the poor performance of the *LDSOnly* measure.

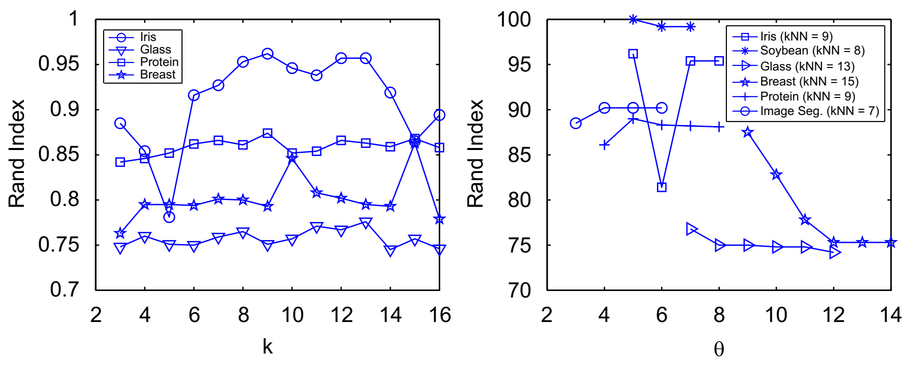


Fig. 3. Left: RI measure of the AHCC+Proposed ASC using 50 queries versus the number of neighbors k in the k-NNG. Right: RI measure of the AHCC+Proposed ASC versus the θ parameter (k is fixed experimentally).

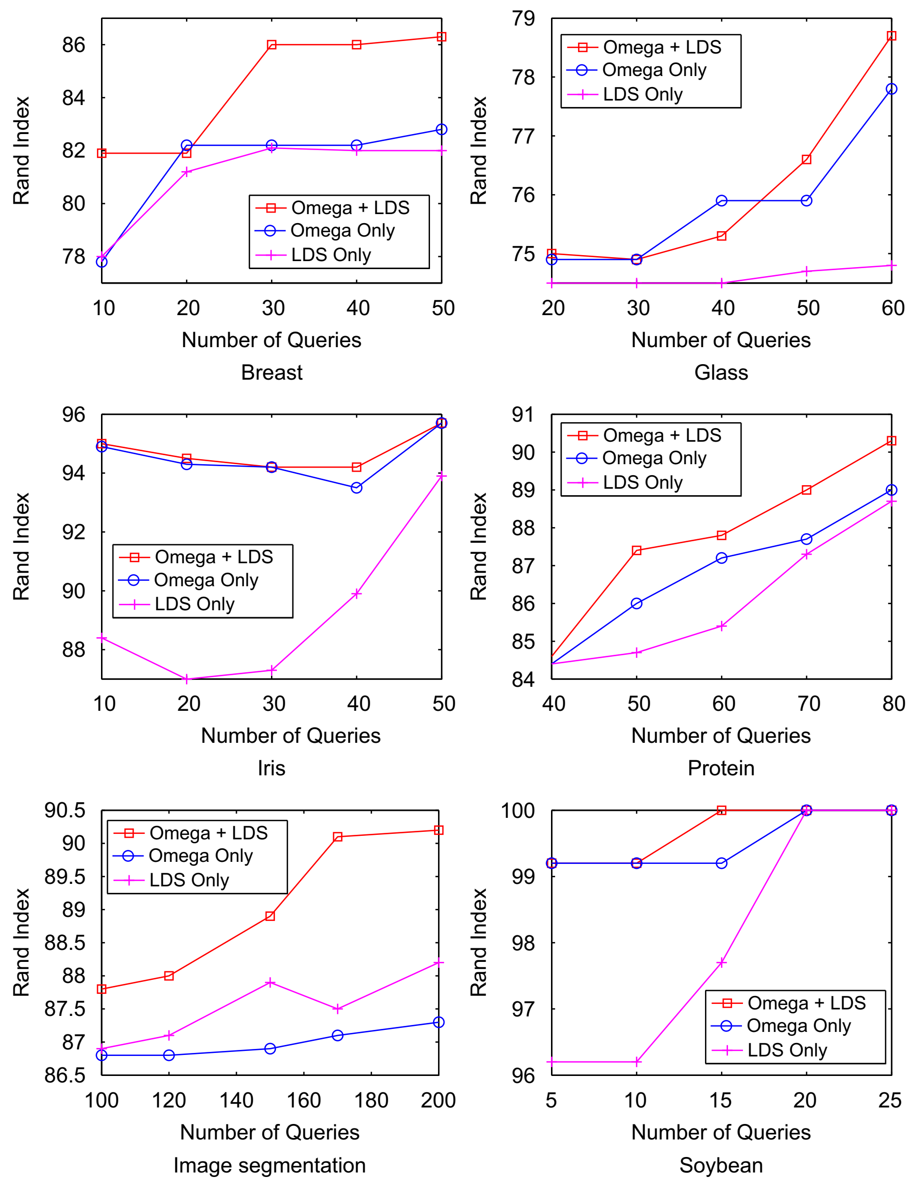


Fig. 4. Comparison of contributions of ω (OmegaOnly) and *LDS* (LDSOnly) compared to the *ASC* measure (Omega + LDS) in terms of clustering error.

However, in general, the *ASC* measure benefits from both indicators ω and *LDS* and thus provides the best clustering results.

4.4. Comparative experiments

Figs. 5 and 6 present the results of the evaluation of our new constraint selection methods "Proposed ASC" (Proposed ASC prod) and "Proposed Random" compared to the Min–Max FFQS heuristic and the "Random" selection of the constraints. In order to evaluate the ability of our proposed approaches to adapt to distinct clustering algorithms, the heuristics are compared with either the AHCC or the MPC-KMeans algorithms. Furthermore, the Figs. 5 and 6 also present the RI score of a classical K-Means clustering algorithm on our benchmark datasets as a baseline. For each dataset and each number of queries the results of the non-deterministic heuristics are averaged over 50 runs.

It can be observed from Figs. 5 and 6 that our approach "Proposed ASC" generally performs better on average for both clustering algorithms (AHCC and MPC-KMeans) than the four other methods even if for some datasets and some number of queries the "Proposed Random" approach may obtain even better results on some runs (as pointed out by the standard deviations). Similarly the "Proposed ASC prod" has better results than the "Proposed ASC" on the Iris dataset because it gives more weight to the *LDS* term which performs well in this case (see Fig. 4). The MMFFQS obtains better results with MPC-KMeans only for the Breast dataset and for Iris dataset when the number of user queries exceeds 50.

This can be explained by the fact that for simpler datasets with a small number of clusters like Breast, the MMFFQS approach generally better manages to define the skeleton of CL constraints during its exploration step, which in turn promotes the initialization of the centers in MPC-KMeans. In the case of more complex datasets like Image Segmentation (seven clusters, 2100 objects) the MMFFQS approach needs more constraints than the "Proposed ASC" to reach an equivalent RI score.

More generally, it can be seen from Figs. 5 and 6, that our "Proposed ASC" heuristic obtains better results than the MMFFQS with small number of queries. For example, in the case of the Soybean dataset, although both "Proposed ASC" and MMFFQS reach 100% accuracy after 20 user queries, it must be noticed that this score drops to 93.5% for MMFFQS with only 10 queries, while

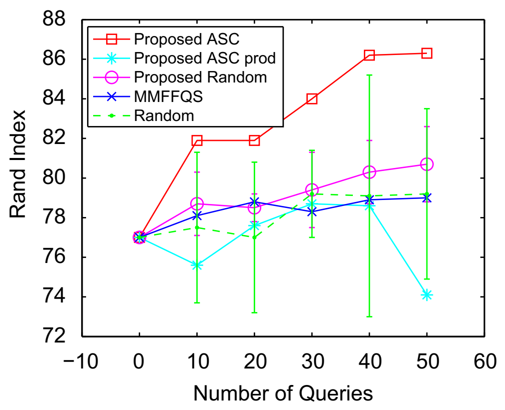
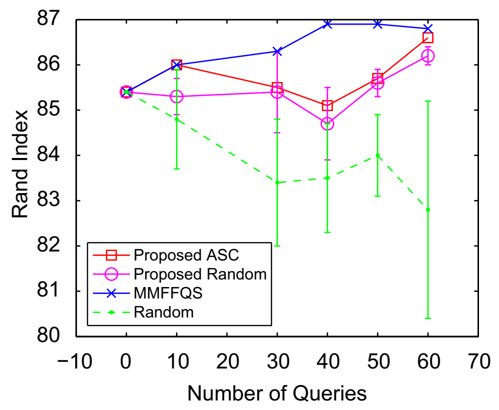
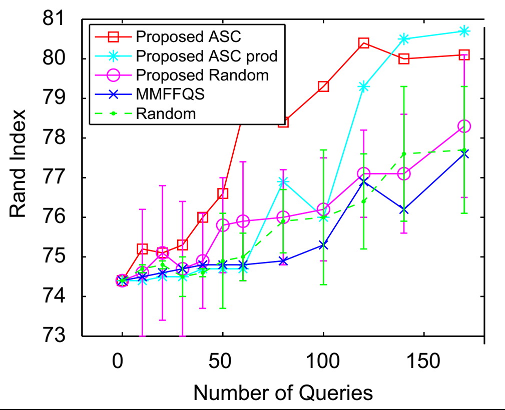
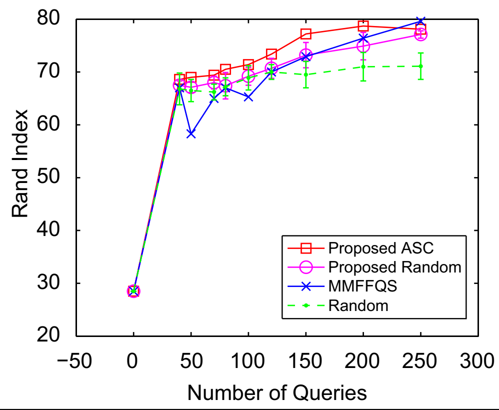
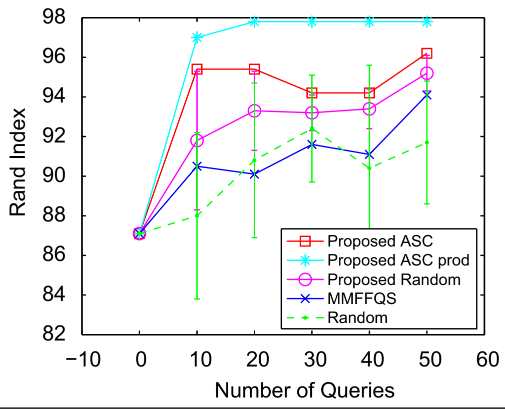
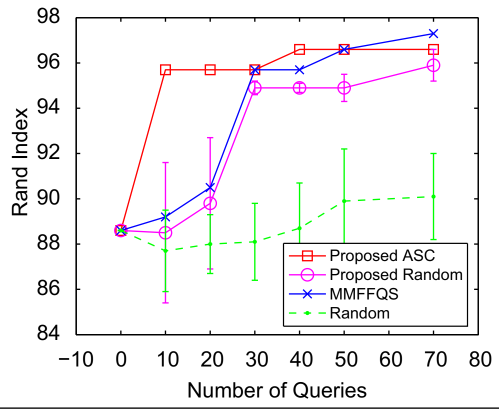
Data	K-Means	Results of AHCC	Results of MPC-K Means
Breast	75.0 ± 0.1		
Glass	68.7 ± 1.5		
Iris	83.2 ± 7.3		

Fig. 5. Comparison of the constraint selection heuristics according to the clustering algorithms (AHCC or MPC-KMeans).

our "Proposed ASC" method still achieves 99.2% accuracy with AHCC. Similarly, with MPC-KMeans, the RI score of the "Proposed ASC" reaches 95.7% after only 10 user queries on Iris dataset and 100% for only 2 user queries on Soybean dataset while MMFFQS gets lower RI score in these cases.

The "Proposed Random" heuristic is generally the second best approach after our "Proposed ASC" when paired with AHCC algorithm, but has similar or slightly worst performances than the MMFFQS when it is paired with the MPC-KMeans algorithm. More generally, our proposed heuristics based on a *k*-NNG perform better comparatively to the other constraint selection heuristics (MMFFQS and "Random") with the AHCC algorithm than with the MPC-KMeans algorithm. This can be explained by the fact that both "Proposed ASC" and "Proposed Random" heuristics promote the selection of constraints at the frontier of the clusters which, on the one hand, helps the hierarchical clustering algorithm AHCC to merge correctly the clusters but, on the other hand, does not allow the MPC-KMeans to produce the best initial partition. Inversely, as already mentioned, the MMFFQS produces constraints in its exploration step that help the initialization process of K-Means like approaches and thus is more efficient with the MPC-KMeans method.

It is also interesting to notice that the clustering performance may decrease locally for some datasets when the number of queries increases, while it is expected that the performance increases monotonically with the number of queries. This problem is a well-known issue in constraint-based clustering that has been addressed in [21] and may be due to either the variability of the random approaches in some cases or due to a bad selection of some constraints that leads constraint based clustering algorithm to poorer clustering results.

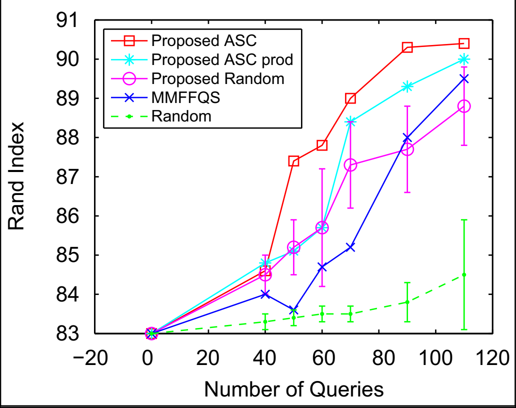
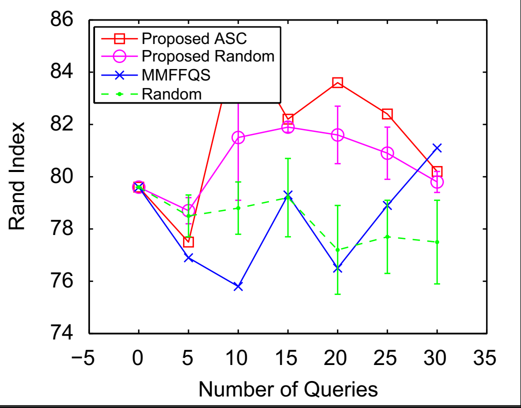
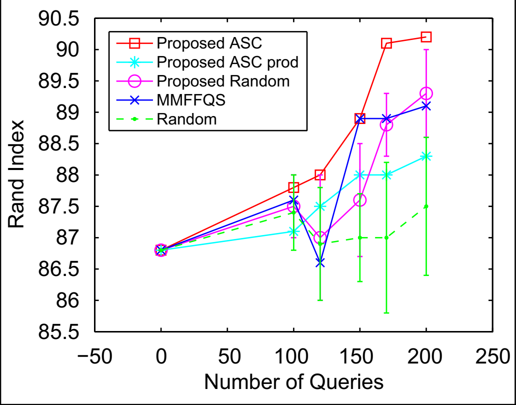
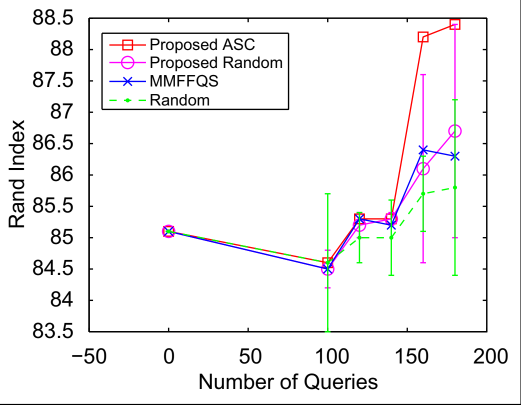
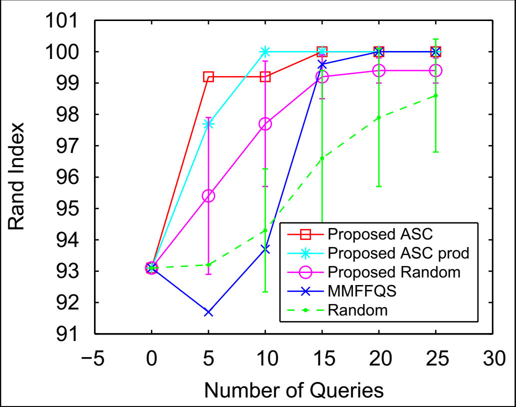
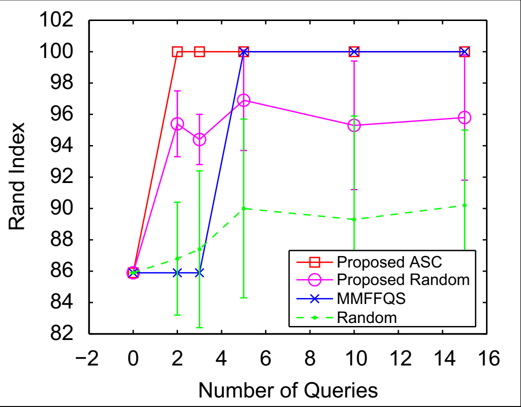
Data	K-Means	Results of AHCC	Results of MPC-KMeans
Protein	76.9 ± 1.4		
Image	82.3 ± 1.8		
Soybean	84.1 ± 6.1		

Fig. 6. Comparison of the constraint selection heuristics according to the clustering algorithms (AHCC or MPC-KMeans).

Figs. 5 and 6 also show that the constraint based clustering algorithms generally outperform the traditional K-Means algorithm except for the MPC-KMeans algorithm with the Glass dataset when the number of user queries is below 50 and for the Protein dataset with the MMFFQS heuristic. This tends to prove the utility of constraint based clustering algorithms over unsupervised approaches when expert knowledge is available.

Finally, these experiments show that the constraints proposed by our active selection process based on the ASC measure are generally more beneficial for both constraint-based clustering algorithms than those provided by the MMFFQS approach and that our "Proposed ASC" heuristic allows to achieve high quality clustering while minimizing the number of user queries compared to the MMFFQS heuristic. Furthermore, as opposed to our "Proposed ASC" heuristic, MMFFQS needs to be initialized with the number of clusters which may limit its usage in some real world applications.

4.5. *Impact of the propagation mechanism*

Fig. 7 presents the number of constraints that are propagated by our algorithm versus the number of queries asked to the users for two representative datasets. It is important to notice that, in the "Random" and MMFFQS methods, each selection of a pair of objects simulates one user query whose answer corresponds to exactly one constraint, whereas in the heuristics "Proposed ASC and Random" each query can lead to several constraints according to the *Propagation* procedure. Fig. 7 shows that the "Proposed Random" heuristic propagates more constraints than the "Proposed ASC" method. However, as our previous results show that the "Proposed ASC" heuristic performs better than the others in term of clustering quality, this indicates that each query generated by the "Proposed ASC" heuristic lead to more beneficial constraints for the clustering algorithm.

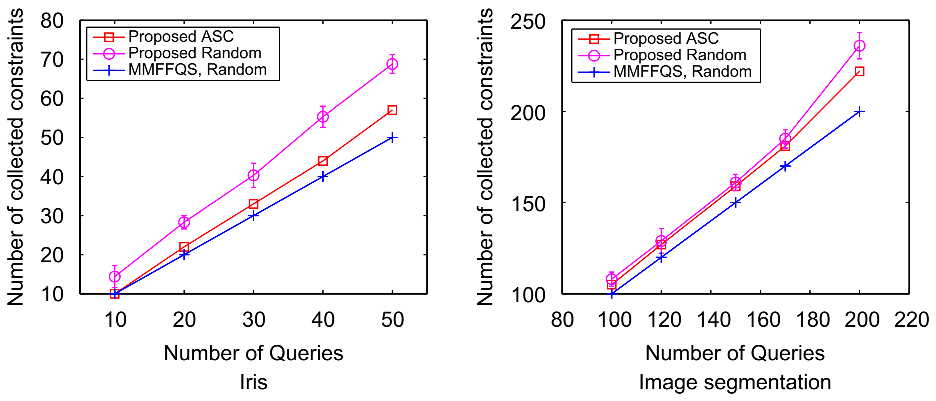


Fig. 7. Number of collected constraints versus number of user queries for Iris and Image segmentation datasets.

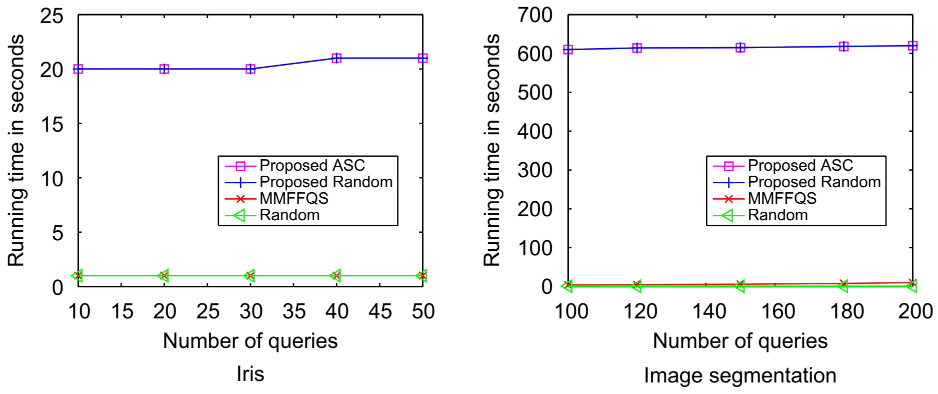


Fig. 8. Average running times for Iris and Image Segmentation datasets.

4.6. *Efficiency of the proposed approaches*

Our main objective is to propose a novel constraint selection approach that improves the clustering quality of constrained-based clustering algorithms while minimizing the human expert effort required to label the constraints. As we do not focus on computation efficiency, our algorithm uses a brute force k-nearest neighbors search method which has a quadratic complexity $O(n^2)$ with the number n of objects in the dataset. Fig. 8 reports the average running times for two representative datasets.

The results show that, as expected, our "Proposed ASC" method is slower than the MMFFQS because of the initial building of the k-nearest neighbors graph. However, the method is still usable on the Image Segmentation dataset that contains 2100 points. As our heuristic gives promising results and performs better than MMFFQS in term of clustering quality and number of user interactions, further studies should be conducted on larger datasets to validate our approach. Furthermore, it could be interesting to reduce the complexity of our algorithm with approximate nearest neighbors search algorithms or dedicated data structure (kd-tree, r-tree, etc.) when the dimensionality of the dataset is low.

5. Conclusion and future works

A new active query selection framework for constrained clustering algorithms is proposed. Contrary to other approaches, our method aims at generating constraints that are beneficial for all kind of constraint based clustering algorithms like constrained K-Means, constrained Fuzzy C-Means, constrained DBSCAN, etc. The novelty of the proposed method relies on three aspects: (1) a k-nearest neighbor graph is used to determine the best candidate queries in the sparse regions of the dataset between the clusters where traditional clustering algorithms perform poorly, (2) a new measure of constraint utility function is used in the queries selection process and (3) a propagation procedure allows each user query to generate several constraints which limits the number of user interactions.

The experiments conducted with two distinct clustering algorithms (AHCC and MPC-KMeans) on real datasets show that our approaches based on a k-NNG outperform in term of clustering quality the random constraint selection approach that is generally used for constraint based clustering algorithms evaluation and also the MMFFQS approach.

Future works include the analysis of the dynamic of propagation of constraints, the test of other aggregation methods between the ω and *LDS* criteria in the *ASC* score and the evaluation of alternative methods to compute the k-nearest neighbors graph such as dedicated data structures (like kd-trees or r-trees) or approximate nearest neighbors algorithms. In a more general point of view, there are still open problems relative to semi-supervised clustering. It could be interesting to work on the problem of datasets that contains overlapping clusters as it has been observed that constraints selected in overlapping regions may decrease the overall clustering quality. Research could also be conducted to introduce other expert knowledge than constraints in clustering algorithms (like cluster seeds [40]). The problem of determining the optimal number of user queries in active learning is still open: in [41] the authors indicate that the decision to continue or to stop the user queries mainly depends on external criterion (like economical factors). Finally, the problem of active constraint selection for other domains like classification and feature selection should also be studied.

References

[1] K.L. Wagstaff, C. Cardie, S. Rogers, S. Schroedl, Constrained *k*-means clustering with background knowledge, in: Proceedings of the 18th International Conference on Machine Learning, ICML-2001, 2001, pp. 577–584.
[2] R. Yan, J. Zhang, J. Yang, A. Hauptmann, A discriminative learning framework with pairwise constraints for video object classification, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (4) (2006) 578–593.
[3] N. Nguyen, R. Caruana, Improving classification with pairwise constraints: a margin-based approach, in: Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML-PKDD-2008, 2008, pp. 113–124.
[4] D. Zhang, S. Chen, Z.-H. Zhou, Constraint score: a new filter method for feature selection with pairwise constraints, Pattern Recognition 41 (5) (2008) 1440–1451.
[5] D. Klein, S.D. Kamvar, C.D. Manning, From instance-level constraints to space-level constraints: making the most of priori knowledge in data clustering, in: Proceedings of the 22nd International Conference on Machine Learning, ICML-2002, 2002, pp. 307–314.
[6] E.P. Xing, A.Y. Ng, M.I. Jordan, S.J. Russell, Distance metric learning with application to clustering with side-information, in: Proceedings of the Conference on Neural Information Processing Systems, NIPS-2002, 2002, pp. 505–512.
[7] M. Bilenko, S. Basu, R.J. Mooney, Integrating constraints and metric learning in semi-supervised clustering, in: Proceedings of the 21st International Conference on Machine Learning, ICML-2004, 2004, pp. 294–307.
[8] A. Bar-Hillel, T. Hertz, N. Shental, D. Weinshall, Learning a Mahalanobis metric from equivalence constraints, Journal of Machine Learning Research 6 (2005) 937–965.
[9] B. Yan, C. Domeniconi, An adaptive kernel method for semi-supervised clustering, in: Proceedings of the European Conference on Machine Learning, ECML-2006, 2006, pp. 521–532.
[10] Y. Liu, R. Jin, A.K. Jain, Boostcluster: boosting clustering by pairwise constraints, in: Proceedings of the 13rd ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD-2007, 2007, pp. 450–459.
[11] D.-Y. Yeung, H. Chang, A kernel approach for semisupervised metric learning, IEEE Transactions on Neural Networks 18 (1) (2007) 141–149.
[12] X. Yin, S. Chen, E. Hu, D. Zhang, Semi-supervised clustering with metric learning: an adaptive kernel method, Pattern Recognition 43 (4) (2010) 1320–1333.
[13] I. Davidson, S.S. Ravi, Clustering with constraints: feasibility issues and the *k*-means algorithm, in: Proceedings of the SIAM International Conference on Data Mining, SDM-2005, 2005, pp. 138–149.
[14] I. Davidson, S.S. Ravi, Agglomerative hierarchical clustering with constraints: theoretical and empirical results, in: Proceedings of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML PKDD-2005, 2005, pp. 59–70.
[15] C. Ruiz, M. Spiliopoulou, E. Menasalvas, C-DBSCAN: density-based clustering with constraints, in: Proceedings of the International Conference on Rough Sets Fuzzy Sets Data Mining and Granular Computing, RSFDGrC-2007, 2007, pp. 216–223.
[16] N. Grira, M. Crucianu, N. Boujemaa, Active semi-supervised fuzzy clustering, Pattern Recognition 41 (5) (2008) 1834–1844.
[17] V.-V. Vu, N. Labroche, B. Bouchon-Meunier, Leader ant clustering with constraints, in: Proceedings of the 7th IEEE RIVF International Conference on Computing and Communication Technologies, RIVF-2009, 2009, pp. 577–584.
[18] X. Wang, I. Davidson, Flexible constrained spectral clustering, in: Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining KDD-2010, 2010, pp. 563–572.
[19] I. Davidson, S. Basu, A survey of clustering with instance level constraints, ACM Transactions on Knowledge Discovery from Data (2007) 1–41.
[20] K.L. Wagstaff, Value, cost, and sharing: open issues in constrained clustering, in: Proceedings of the 5th International Workshop on Knowledge Discovery in Inductive Databases, KDID-2007, 2007, pp. 1–10.
[21] I. Davidson, K.L. Wagstaff, S. Basu, Measuring constraints-set utility for partitional clustering algorithms, in: Proceedings of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases ECML, PKDD-2006, 2006, pp. 115–126.
[22] P.K. Mallapragada, R. Jin, A.K. Jain, Active query selection for semi-supervised clustering, in: Proceedings of the International Conference on Pattern Recognition, ICPR-2008, 2008, pp. 1–4.
[23] A.K. Jain, Data clustering: 50 years beyond *k*-means, Journal of Pattern Recognition Letters 31 (8) (2010) 651–666.
[24] S. Basu, I. Davidson, K.L. Wagstaff, Constrained Clustering: Advances in Algorithms, Theory, and Applications, Data Mining and Knowledge Discovery Series, 1st ed., Chapman and Hall/CRC, 2008.
[25] S. Basu, A. Banerjee, R.J. Mooney, Active semi-supervision for pairwise constrained clustering, in: Proceedings of the SIAM International Conference on Data Mining, SDM-2004, 2005, pp. 333–344.
[26] V.-V. Vu, N. Labroche, B. Bouchon-Meunier, An efficient active constraint selection algorithm for clustering, in: Proceedings of the 20th International Conference on Pattern Recognition ICPR-2010, 2010, pp. 2969–2972.
[27] V.-V. Vu, N. Labroche, B. Bouchon-Meunier, Boosting clustering by active constraint selection, in: Proceedings of the 19th European Conference on Artificial Intelligence, ECAI-2010, 2010, pp. 297–302.
[28] S.D. Kamvar, D. Klein, C.D. Manning, Spectral learning, in: Proceedings of the Joint Conference on Artificial Intelligence, IJCAI-2003, 2003, pp. 561–566.
[29] Q. Xu, M. desJardins, K.L. Wagstaff, Active constrained clustering by examining spectral eigenvectors, in: Proceedings of Discovery Science Conference, DS-2005, 2005, pp. 294–307.
[30] R.A. Jarvis, E.A. Patrick, Clustering using a similarity measure based on shared near neighbors, IEEE Transactions on Computer 22 (11) (1973) 1025–1034.
[31] D.-D. Le, S. Satoh, Unsupervised face annotation by mining the web, in: Proceedings of the IEEE International Conference on Data Mining, IEEE ICDM-2008, 2008, pp. 383–392.
[32] C.L. Blake, C.J. Merz, Uci Machine Learning Repository (source: 〈http://archive.ics.uci.edu/ml/〉), 1998.
[33] D. Lewis, J. Catlett, Heterogeneous uncertainly sampling for supervised learning, in: Proceedings of the 11st International Conference on Machine Learning, ICML-1994, 1994, pp. 148–156.
[34] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of the 2nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD-1996, 1996, pp. 226–231.
[35] X. Zhu, A.B. Goldberg, T. Khot, Some new directions in graph-based semi-supervised learning, in: Proceedings of the IEEE International Conference on Multimedia and Expo, IEEE ICME-2009, 2009, pp. 1504–1507.
[36] M. Breunig, H.-P. Kriegel, R. Ng, J. Sander, Lof: identifying density-based local outliers, in: Proceedings of the 19th ACM SIGMOD International Conference on Management of Data, VLDB-2000, 2000, pp. 93–104.
[37] R. Weber, H.-J. Schek, S. Blottt, A quantitative analysis and performance study for similarity-search methods in high-dimensional space, in: Proceedings of the ACM International Conference on Very Large Data Bases, VLDB-1998, 1998, pp. 194–205.
[38] A.K. Jain, A. Topchy, M.H.C. Law, J.M. Buhmann, Landscape of clustering algorithms, in: Proceedings of the 17th International Conference on Pattern Recognition, ICPR-2004, 2004, pp. 260–263.
[39] W.M. Rand, Objective criteria for evaluation of clustering methods, Journal of the American Statistical Association 66 (1971) 846–850.
[40] V.-V. Vu, N. Labroche, B. Bouchon-Meunier, Active learning for semi-supervised *k*-means clustering, in: Proceedings of the 22nd International Conference on Tools with Artificial Intelligence IEEE, ICTAI-2010, 2010, pp. 12–15.
[41] B. Settles, Learning Literature Survey, 2010.

Viet-Vu Vu received the B.S. degree in Computer Science from Ha Noi University of Education in 2000 and a M.S. degree in Computer Science from Ha Noi University of Technology in 2004. He is currently pursuing the Ph.D. degree in the Laboratory of Computer Science (LIP6) at the Pierre and Marie Curie University (Paris 6). His research interests include clustering, active learning, and semi-supervised clustering.

Nicolas Labroche is an Associate Professor of Computer Science at University Pierre and Marie Curie - UPMC Paris 6, France. He received a Ph.D. degree in Computer Sciences in 2003 and a M.S. degree in Computer Science and a M.S. degree in Biological and Medical Signal and Image Processing in 2000. His research interests include clustering, pattern recognition, web usage analysis, biomimetic approaches.

Bernadette Bouchon-Meunier is a director of research at the National Centre for Scientific Research, head of the department of Databases and Machine Learning in the Computer Science Laboratory of the University Paris 6. Graduate from the Ecole Normale Superieure at Cachan, she received the degrees of B.S. in Mathematics and Computer Science, Ph.D. in Applied Mathematics and D.Sc. in Computer Science from the University of Paris.

Editor-in-Chief of the International Journal of Uncertainty, Fuzziness and Knowledge-based Systems (World Scientific), she is also the (co)-editor of 22 books and the (co)-author of five books on uncertainty management in artificial intelligence. Co-executive director of the International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems (IPMU), she also served as the FUZZ-IEEE 2010 Program Chair and she is the General Chair of the IEEE Symposium Series on Computational Intelligence SSCI 2011. She is an IEEE fellow and an International Fuzzy Systems Association fellow.

Her present research interests include approximate reasoning and applications of fuzzy logic and machine learning techniques to decision-making, data mining, risk forecasting, information retrieval and user modeling.