ECAI 2010
H. Coelho et al. (Eds.)
IOS Press, 2010

doi:10.3233/978-1-60750-606-5-297

Boosting Clustering by Active Constraint Selection

Viet-Vu Vu, Nicolas Labroche, and **Bernadette Bouchon-Meunier** [1]

Abstract. In this paper we address the problem of active query selection for clustering with constraints. The objective is to determine automatically a set of user queries to define a set of must-link or cannot-link constraints. Some works on active constraint learning have already been proposed but they are mainly applied to K-Means like clustering algorithms which are known to be limited to spherical clusters, while we are interested in clusters of arbitrary sizes and shapes. The novelty of our approach relies on the use of a k-nearest neighbor graph to determine candidate constraints coupled with a new constraint utility function. Comparative experiments conducted on real datasets from machine learning repository show that our approach significantly improves the results of constraints based clustering algorithms.

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 because these methods allow to take into account a user's knowledge (called *oracle* or *teacher* in this case) - 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 [25]. ML constraints indicate that two points of the dataset have to be partitioned in the same cluster while CL constraints impose that the points belong to different clusters.

We can divide previous work on clustering with constraints into two main families: either 1) the constraints help the algorithm to learn a metric/objective function [3, 8, 15, 21, 18, 6, 20] or 2) the constraints are used as *hints* to guide the algorithm to a useful solution [7, 23, 25, 22].

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 [24]?* The need for a constraint set utility measure has become imperative with the recent observation that some poorly defined constraint sets can decrease clustering performances [9, 24]. We will propose a new measure to evaluate a constraint utility. This measure evaluates the ability of a constraint to help the clustering algorithm to distinguish the points in the perturbation regions, e.g. sparse regions or transition regions. We use this measure to develop an active constraint selection algorithm.
2. *How can we minimize the effort required to the user, by only soliciting her(him) for the most useful constraints [13, 24]?* Many researches have been conducted on the problem of clustering with constraints [3, 7, 22, 23, 18, 2, 6, 20, 12] but most of the time the user is supposed to provide the algorithm with good constraints in a passive manner (see Figure 1). 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 [9] and as there is possibly $\frac{n\times(n-1)}{2}$ ML or CL constraints in a datasets 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 [1, 19].

This paper presents a new active constraint selection algorithm to collect a constraint set which can be suitable for constrained clustering algorithms that apply to clusters with different sizes and arbitrary shapes (Constrained-DBSCAN [22], Constrained Hierarchical Clustering [7], and Constrained Spectral Clustering [27]). Our method relies on a k-nearest neighbor graph to estimate sparse regions of the data where queries about constraints are most likely to be asked.

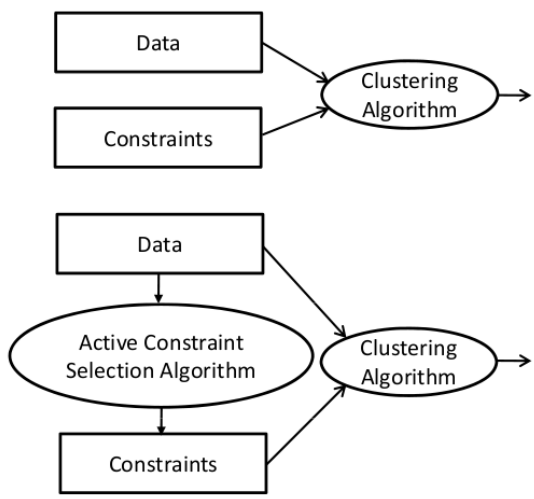

Figure 1. Illustration of passive definition of constraints (top) and active constraints learning (bottom)

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 on benchmark datasets. Finally, section 5 concludes and discusses future research.

2 RELATED WORKS

There are few works on active constraint selection for clustering. In [1], an algorithm for active constraint selection for K-means using farthest-first strategy was proposed. This algorithm is referred to as

[1] Université Pierre et Marie Curie - Paris 6, CNRS UMR 7606, LIP6, Paris, France, email: {viet-vu.vu, nicolas.labroche, bernadette.bouchon-meunier}@lip6.fr

the Farthest First Query Selection (FFQS) [19]. 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; 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 a ML constraint is obtained from the user.

In [19], an improved version of FFQS using a min-max approach is proposed. In the *Consolidate* phase, instead of randomly selecting the point, the idea consists of selecting the data point whose largest similarity to the *skeleton* is the smallest. By this way, the data point with largest uncertainty in cluster membership is chosen earliest to express the queries. However, both previous methods do not work well in the case of a data set with a large number of clusters or unbalanced data sets with small clusters.

Finally, in [27], 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-cluster 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 Neighbor Graph (*k*-NNG). Second, our approach makes use of a new constraint set utility measure to rank the candidate constraints according to their ability to separate clusters.

3.1 The k-nearest neighbor 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 points, x_i and x_j, if and only if x_i and x_j have each other in their k-nearest neighbors set. The weight $\omega(x_i, x_j)$ of the edge (the similarity) between two points x_i and x_j is defined as the number of common nearest neighbor the two points share, as shown in equation 1 [10].

$$\omega(x_i, x_j) =\mid NN(x_i) \cap NN(x_j) \mid \quad (1)$$

where *NN(.)* denotes the set of k-nearest neighbors of the specified 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 constraints utility

Given a *k*-NNG, the Local Density Score (*LDS*) of a vertex $x_i \in$ *k*-NNG is defined by equation 2 [16]:

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

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 x_i and its neighbors, i.e. x_i belongs to a dense region. In contrast, a low value of *LDS* (high value of $\frac{1}{1+LDS}$) means that x_i belongs to a sparse region or transition region between clusters. In addition, in the k-NNG defined above, edges with small weight values of ω (high value of $(k-\omega)$) are also in the sparse regions or transition regions.

From these properties, we see that a constraint will be valuable if it can help the clustering algorithms to separate the points in the sparse regions or transition regions, that is to say where it is difficult for the algorithms like DBSCAN [11] or Hierarchical clustering to distinguish between different clusters. We define a new utility measure of a constraint between two points x_i and x_j as its Ability to Separate between Clusters (*ASC*) as shown in equation 3:

$$ASC(x_i, x_j) = \frac{k - \omega(x_i, x_j) + \frac{1}{1+min\{LDS(x_i), LDS(x_j)\}}}{k+1} \quad (3)$$

The utility of a constraint depends on two aspects: the weight $\omega(x_i, x_j)$ and the density of the region it belongs to. We define the density of a constraint between two points x_i and x_j as the minimum between $LDS(x_i)$ and $LDS(x_j)$ which means that if one of the two points is in sparse region then the density of the constraint is likely to be small.

The value of *ASC* ranges in $[\frac{1}{k}, 1]$. We make the hypothesis that a constraint with high *ASC* score is more likely to provide performance gains.

3.3 Identification of candidate constraints

Following the principle of active learning [17], 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 these cases, the cluster membership is the most uncertain. In order to limit the number of constraint candidates for the set of candidates C (see equation 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.

$$C = \{(u, v) \mid weight(u, v) < \theta\} \quad (4)$$

To generate the user queries, we can randomly choose the candidates from *C*, or we can rank the candidates in descending order according 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 first 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 condition in the section 3.3 and asks the user about the nature of the relation: ML, 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).

Given a set of candidate constraints C and a set of constraints Y, the **Propagation** procedure aims at discovering new constraints in C from the information stored in Y. We first need to define the notion of *strong path*.

Definition 1 *Given a k-nearest neighbor graph (k-NNG) for a data set X and a threshold θ, a path from vertex u to vertex v is defined as strong path $SP(u, v, \theta)$ iff there exists a sequence of vertex $(z_1, z_2, ..., z_t)$ such that $u = z_1$, $v = z_t$ and $\forall i = 1 \dots t-1$: $\omega(z_i, z_{i+1}) \geq \theta$.*

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

1. 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 Figure 2).
2. given two constraints (u, v) and (v, w) in Y, we have [19]: (i) $ML(u, v) \wedge ML(v, w) \Rightarrow ML(u, w)$; (ii) $ML(u, v) \wedge CL(v, w) \Rightarrow CL(u, w)$. These new generated constraints are added to Y.

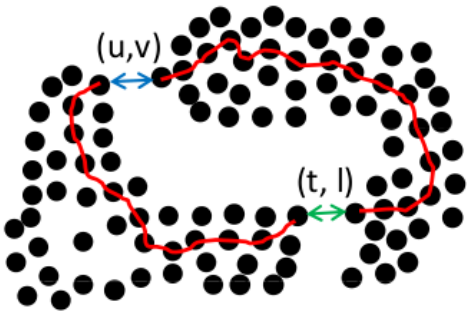

Figure 2. Illustration of the propagation mechanism: $Label(t, l)$ is propagated as $Label(u, v)$ because of the presence of strong paths $SP(u, t, \theta)$ and $SP(v, l, \theta)$ between them

Finally, the **Refinement** procedure is called to filter candidates that could be linked by a *strong path*: all the edges (u, v) such that there exists a strong path $SP(u, v, \theta)$ between u and v are removed from the candidate constraints set. The objective is to identify points that are indirectly connected through a dense region (a similar principle is found in the problem of propagation of labeled data for semi-supervised classification [28]). This procedure is crucial for the performance of our approach since it allows to decrease considerably the size of the candidate constraints set.

The main steps of our algorithm are summed up in **Algorithm 1** and **Algorithm 2**.

The complexity of our algorithm depends on the complexity of the building of a k-NNG and the **Refinement** procedure. The complexity of building the k-NNG is $O(n$*time for a k-nearest neighbors query). Following Breunig *et al.* [5], for k-nearest neighbors queries, we have a choice among different methods. For low-dimensional data, we can use a grid based approach which can answer k-nearest neighbors queries in constant time, leading to a complexity of $O(n)$ to build the k-NNG. For medium to medium high-dimensional data, we can use an index, which provides an average complexity of $O(logn)$ for k-nearest neighbors queries, leading to a complexity of $O(n * logn)$ to build the k-NNG. For extremely high-dimensional data, we need to use a sequential scan or some variant of it, e.g. the VA-file [26], with a complexity of $O(n)$, leading to a complexity of $O(n^2)$ to build the k-NNG. The complexity of **Refinement** procedure is $O(n * k)$ (scan all the edges of k-NNG).

Algorithm 1 Active constraint selection

Input: *Data set $X = \{x_i\}_{i=1}^n$, k and threshold θ*
Output: *Set of collected constraints Y*
Process:
$Y = \emptyset$
Construct a k-NNG of X
$C = \{(u, v) \mid weight(u, v) < \theta\}$
Refinement(C, θ)
while (UserStop = False) **and** $(C \neq \emptyset)$ **do**
  Pick a $(u, v) \in C$ following the section 3.3
  Query the user about the $Label$ of (u, v) ?
  if $Label$ is *ML* **or** *CL* **then**
    $Y = Y \cup \{Label(u, v)\}$
    Propagation(C, Y, θ)
    Refinement(C, θ)
  else
    Save (u, v) and that query is not asked again later
  end if
end while

Algorithm 2 Refinement(C, θ)

Process:
for all (u,v) $\in$ C **do**
  if $\exists$ SP(u,v, θ) **then**
    $C = C - \{(u, v)\}$
  end if
end for

So, the complexity of our algorithm is $O(n * k)$, $O(n * logn)$ or $O(n^2)$ when the dimension of data is respectively low, medium or extremely high.

4 EXPERIMENT RESULTS

4.1 Experimental Protocol

We use 8 real datasets from the Machine Learning Repository [4] named: Iris, Soybean, Wine, Pima, Glass, Spectf, Ecoli, and Breast to evaluate our algorithm. The detail of datasets are shown in Table 1.

We use the Agglomerative Hierarchical Clustering with Constraints [7] algorithm to evaluate the efficiency of our active selection constraints framework. We refer to this algorithm as AHCC algorithm. We also note that the AHCC is one of the three important types of clustering algorithm in practice according to the research of Jain *et al.* [14]. AHCC inputs a set of constraints (ML and CL) and returns a dendrogram which satisfies all the constraints. For each dendrogram, we choose the best output partition for the evaluation of results. Naturally, our framework can be easily adapted to other constraint-based clustering algorithm like Constrained-DBSCAN, constrained spectral clustering ...

4.2 Evaluation method

The data set used for the evaluation includes a "correct answer" or label for each data point. We use the labels in a post-processing step to evaluate the performance of our approach.

We use the Rand Index (*RI*) measure [25], as it is widely used in evaluation of clustering results.

Table 1. Main characteristics of the real datasets

ID	Data	#Objects	#Attributes	#Clusters
1	Iris	150	4	3
2	Soybean	47	35	4
3	Wine	178	13	3
4	Pima	768	8	2
5	Glass	214	9	6
6	Spectf	267	22	2
7	Ecoli	336	8	8
8	Breast	569	30	2

The *RI* measure computes the agreement between the theoretical partition of each dataset and the output partition of evaluated algorithms. This measure is based on $\frac{n(n-1)}{2}$ pairwise comparisons between the n points of a data set X. For each pair of points x_i and x_j in X, a partition assigns them either to the same cluster or to different clusters.

Let us consider two partitions P_1 and P_2, and let a be the number of decisions where the point x_i is in the same cluster as x_j in P_1 and P_2. Let b be the number of decisions where the two points are placed in different clusters in both partitions. A total agreement can then be calculated as shown in equation 5.

$$RI(P_1, P_2) = \frac{2(a+b)}{n(n-1)} \tag{5}$$

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 Choosing the parameters

Our active selection framework uses two parameters: the number of nearest neighbors k and the threshold θ. As shown in Figure 3, the value of k cannot be generalized for all datasets because it depends on the structure and the size of the datasets. For example, in the Iris dataset, the changes in the value of k to some local optimum that are due to overlapping clusters. However, we have observed experimentally that, for all datasets, the best results are obtained when parameter θ is fixed in $[\frac{k}{2} - 2, \frac{k}{2} + 2]$.

4.4 Results

To evaluate our active query selection algorithm and our measure of constraint utility *ASC*, we used the three following algorithms:

1. AHCC with our active constraint selection algorithm in which the candidates are selected according to the *ASC* score (see section 3.3). This method is deterministic, so we only need to perform it once.
2. AHCC with our active constraint selection algorithm, but the candidates are randomly chosen from candidates set to ask the users. We conducted 50 runs for each dataset and the results were averaged.
3. AHCC with random selection of constraints from data, this method generates a set of ML and CL constraints based on the comparison of the labels of randomly chosen pairs of objects. If two labels are in the same cluster, we generate a ML constraint, and else, we generate a CL constraint. As this approach is non deterministic, the results are averaged over 50 trials.

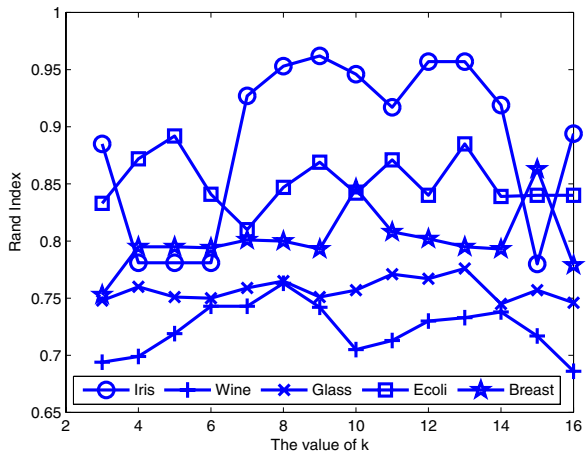

Figure 3. Rand Index measure of our approach with ASC score for some datasets using 50 queries vs. the number of neighbors k in the k-NNG

Figure 4 shows that our method based on the *ASC* measure generally performs better than the method based on a random choice of constraints, which shows the usefulness and the efficiency of the *ASC* measure. However, some complementary observations can be made from these results.

In the case of Pima and Spectf data, the performance of the AHCC significantly decreases when adding the constraints (in the case where the constraints are randomly chosen from labels). It can be explained by the fact that when the data set consists in multiple overlapping clusters, the performance of the constrained clustering algorithms may decrease when constraints are not properly chosen.

In the case of Ecoli dataset for 20 queries, the random generated constraints performs better than the *ASC* approach while after, when the number of queries increases, *ASC* approach gives better results. This result may be due to the use of a hierarchical clustering algorithm whose performances can change drastically for a small number of constraints.

The Table 2 presents the number of queries asked to the users and the number of constraints that are propagated by our algorithm. It is important to notice that, in the approach that selects at random the constraints from the data labels, each selection of a pair of objects simulates one user query whose answer corresponds to one constraint, whereas in both other approaches (based on *ASC* score or random selection of candidates from k-NNG) each query can lead to several constraints according to the *Propagation* procedure. Table 2 shows that the second method, in which constraints are chosen randomly from the set of candidates, propagates more constraints than the method based on *ASC* score. However Figure 4 shows that the method based on *ASC* score performs better than the others, which may indicates that this approach generates better candidate constraints.

5 CONCLUSION

A new active query selection framework for constrained clustering algorithm is proposed. Contrary to other approaches, our method aims at generating constraints that are useful for clustering algorithms like C-DBSCAN, that discover clusters of arbitrary shapes

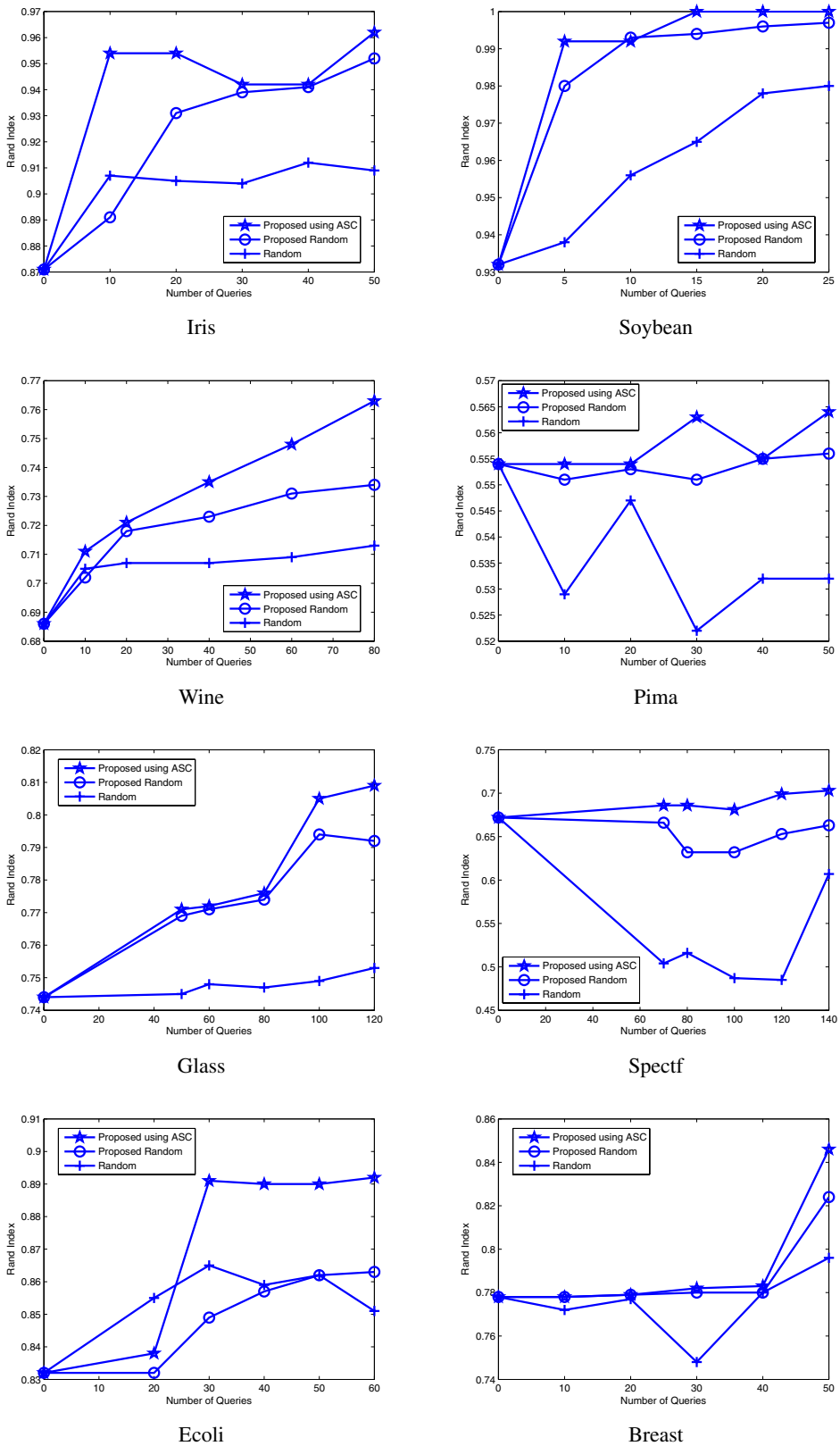

Figure 4. Results of the 8 real datasets using Rand Index measure. As it can be clearly seen, the method based on *ASC* measure performs better than the approach with a random choice of candidates. However, both approaches significantly improve the results compared to the approach with a random choice of candidates from data.

Table 2. Number of queries and number of collected constraints. #C1 is the number of constraints propagated by our algorithm with the *ASC* measure while #C2 is the number (the average value) of constraints propagated with random choice of candidates over 50 trials and σ_{cf} is standard deviation of #C2.

Iris	#Queries	10	20	30	40	60
	#C1	**13**	**26**	**37**	**53**	**81**
	#C2[σ_{cf}]	13[2.2]	24[3.1]	38[3.5]	54.4[3.1]	78[4.7]
Soy-bean	#Queries	10	20	30	40	50
	#C1	**10**	**25**	**42**	**55**	**71**
	#C2[σ_{cf}]	13[2.2]	24[2.1]	41[2.1]	56[2.5]	68[5.1]
Wine	#Queries	10	20	40	60	80
	#C1	**13**	**28**	**56**	**106**	**123**
	#C2[σ_{cf}]	19[3.8]	39[8.4]	78[8.3]	120[9.5]	155[9.8]
Pima	#Queries	10	20	30	40	50
	#C1	**10**	**23**	**33**	**45**	**56**
	#C2[σ_{cf}]	13[2.8]	25[4.0]	38[4.0]	50[4.1]	59[3.5]
Glass	#Queries	10	20	30	40	60
	#C1	**10**	**22**	**34**	**45**	**65**
	#C2[σ_{cf}]	12[1.2]	24[2.5]	37[4.2]	48[1.5]	71[2.9]
Spectf	#Queries	70	80	100	120	140
	#C1	**73**	**84**	**107**	**129**	**151**
	#C2[σ_{cf}]	74[4.2]	87[6.1]	109[3.9]	132[5.5]	154[5.2]
Ecoli	#Queries	20	40	60	80	100
	#C1	**20**	**40**	**61**	**86**	**107**
	#C2[σ_{cf}]	22[1.5]	44[2.6]	68[1.7]	90[2.6]	113[1.5]
Breast	#Queries	10	20	30	40	50
	#C1	**15**	**31**	**44**	**71**	**87**
	#C2[σ_{cf}]	17[3.6]	37[4.8]	45[4.7]	70[4.2]	82[3.9]

and sizes. The novelty of the 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 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. Experiments show that our algorithm outperforms a method based on random queries generation on a set of real datasets, and that the queries and their associated constraints appear to be more suitable for clusters of various shapes and sizes.

Future works include the analysis of the dynamic of propagation of constraints and the development of a visualization interface to improve the interaction between our algorithm and the users. Finally, the problem of active constraint selection when the dataset consists of multiple overlapping clusters will be examined in future researches.

REFERENCES

[1] S. Basu, A. Banerjee, and R.J. Mooney, 'Active semi-supervision for pairwise constrained clustering', in *Proceedings of the SIAM International Conference on Data Mining*, pp. 333–344, (2004).

[2] S. Basu, I. Davidson, and K.L. Wagstaff, *Constrained Clustering: Advances in Algorithms, Theory, and Applications*, Chapman and Hall/CRC Data Mining and Knowledge Discovery Series, 1st edn., 2008.

[3] M. Bilenko, S. Basu, and R.J. Mooney, 'Integrating constraints and metric learning in semi-supervised clustering', in *Proceedings of the 21st International Conference on Machine Learning*, pp. 294–307, (2004).

[4] C.L. Blake and C.J. Merz. Uci machine learning repository, 1998.

[5] M. Breunig, H.-P. Kriegel, R.T. Ng, and J. Sander, 'Lof: Identifying density-based local outliers', in *Proceedings of the 19th ACM SIGMOD International Conference on Management of Data*, pp. 93–104, (2000).

[6] I. Davidson, 'Knowledge driven dimension reduction for clustering', in *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, (2009).

[7] I. Davidson and S.S. Ravi, 'Agglomerative hierarchical clustering with constraints: Theoretical and empirical results', in *Proceeding of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, pp. 59–70, (2005).

[8] I. Davidson and S.S. Ravi, 'Clustering with constraints: Feasibility issues and the k-means algorithm', in *Proceedings of the SIAM International Conference on Data Mining*, (2005).

[9] I. Davidson, K.L. Wagstaff, and S. Basu, 'Measuring constraints-set utility for partitional clustering algorithms', in *Proceeding of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, (2006).

[10] L. Ertoez, M. Steinbach, and V. Kumar, 'Fiding clusters of different sizes, shapes, and densities in noisy, high dimentional data', in *Proceedings of the SIAM International Conference on Data Mining*, (2003).

[11] M. Ester, H.-P Kriegel, J. Sander, and X. Xu, 'A density-based algorithm for discovering clusters in large spatial databases with noise', in *Proceeding of the 2nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, USA, (1996).

[12] E. Fromont, A. Prado, and C. Robardet, 'Constraint-based subspace clustering', in *Proceedings of the SIAM International Conference on Data Mining*, pp. 26–37, (2009).

[13] A.K. Jain, 'Data clustering: 50 years beyond k-means', *Journal of Pattern Recognition Letters*, (2009).

[14] A.K. Jain, A. Topchy, M.H.C. Law, and J.M. Buhmann, 'Lanscape of clustering algorithms', in *Proceedings of the 17th International Conference on Pattern Recognition*, (2004).

[15] D. Klein, S.D. Kamvar, and 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*, (2005).

[16] D.-D. Le and S. Satoh, 'Unsupervised face annotation by mining the web', in *Proceedings of the IEEE International Conference on Data Mining*, (2008).

[17] D.D. Lewis and J. Catlett, 'Heterogeneous uncertainly sampling for supervised learning', in *Proceedings of the 11st International Conference on Machine Learning*, pp. 148–156, (1994).

[18] Y. Liu, R. Jin, and A.K. Jain, 'Boostcluster: Boosting clustering by pairwise cosntraints', in *Proceedings of the 13rd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, (2007).

[19] P.K. Mallapragada, R. Jin, and A.K. Jain, 'Active query selection for semi-supervised clustering', in *Proceedings of the 19th International Conference on Pattern Recognition*, (2008).

[20] Z. Qi and I. Davidson, 'A principled and flexible framework for fiding alternative clusterings', in *Proceedings of the 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, (2009).

[21] K. Rothaus and X. Jiang, 'Constrained clustering by a novel graph-based distance transformation', in *Proceedings of the 19th International Conference on Pattern Recognition*, (2008).

[22] C. Ruiz, M. Spiliopoulou, and 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*, pp. 216–223, (2007).

[23] V.-V. Vu, N. Labroche, and B. Bouchon-Meunier, 'Leader ant clustering with constraints', in *Proceedings of the 7th IEEE RIVF International Conference on Computing and Communication Technologies*, (2009).

[24] K.L. Wagstaff, 'Value, cost, and sharing: Open issues in constrainted clustering', in *Proceeding of the 5th International Workshop on Knowledge Discovery in Inductive Databases*, pp. 1–10, (2007).

[25] K.L. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl, 'Constrained k-means clustering with background knowledge', in *Proceedings of the 18th International Conference on Machine Learning*, pp. 577–584, (2001).

[26] R. Weber, H.-J. Schek, and S. Blottt, 'A quantitative analysis and performance study for similarity-search methods in high-dimentional space', in *Proceedings of the ACM International Conference on Very Large Data Bases*, (1998).

[27] Q. Xu, M. desJardins, and K.L. Wagstaff, 'Active constrained clustering by examining spectral eigenvectors', in *Proceedings of Discovery Science Conference*, pp. 294–307, (2005).

[28] X. Zhu, A.B. Goldberg, and T. Khot, 'Some new directions in graph-based semi-supervied learning', in *Proceedings of the IEEE International Conference on Multimedia and Expo*, (2009).