Density Based Cluster Growing via Dominant Sets

Article *in* Neural Processing Letters · October 2018

DOI: 10.1007/s11063-017-9767-3

3 authors:

Jian Hou
Dongguan University of Technology
99 PUBLICATIONS **1,574** CITATIONS
SEE PROFILE

Emory Xu
Nanyang Technological University
15 PUBLICATIONS **104** CITATIONS
SEE PROFILE

Liu Weixue
Beijing University of Posts and Telecommunications
13 PUBLICATIONS **120** CITATIONS
SEE PROFILE

Neural Processing Letters manuscript No.
(will be inserted by the editor)

Density Based Cluster Growing via Dominant Sets

Jian Hou · Xu E · Weixue Liu

Received: date / Accepted: date

Abstract Although there are a lot of clustering algorithms available in the literature, existing algorithms are usually afflicted by practical problems of one form or another, including parameter dependence and the inability to generate clusters of arbitrary shapes. In this paper we aim to solve these two problems by merging the merits of dominant sets and density based clustering algorithms. We firstly apply histogram equalization to eliminate the parameter dependence problem of the dominant sets algorithm. Noticing that the obtained clusters are usually smaller than the real ones, a density threshold based cluster growing step is then used to improve the clustering results, where the involved parameters are determined based on the initial clusters. This is followed by the second cluster growing step which makes use of the density relationship between neighboring data. Data clustering experiments and comparison with other algorithms validate the effectiveness of the proposed algorithm.

Keywords cluster growing · dominant sets · parameter independence

1 Introduction

Data clustering refers to the process of grouping data into clusters based on their similarity (distance) so that the data in one cluster are similar to each

This work is supported in part by National Natural Science Foundation of China under Grant No. 61473045 and Natural Science Foundation of Liaoning Province under Grant No. 20170540013, and in part by China Scholarship Council.

J. Hou
College of Engineering, Bohai University, Jinzhou 121013, China
ECLT, Università Ca' Foscari Venezia, Venezia 30124, Italy
E-mail: dr.houjian@gmail.com

X. E and W. Liu
College of Information Science, Bohai University, Jinzhou 121013, China

other and they are less similar to those in other clusters. As data clustering is able to find out the hidden pattern among the data, it plays an important role in such various fields as machine learning, data mining, pattern recognition and image analysis. In the past decades, a lot of clustering algorithms have been proposed from different perspectives. Some of the popular algorithms include k-means, DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [10], normalized cuts (NCuts) [30], mean shift [6,7] and their variants. Recent developments in this domain include affinity propagation (AP) [3], dominant sets (DSets) [26], robust spectral clustering [36] and density peak (DP) based algorithm [27].

Although various clustering algorithms can be found in the literature, existing algorithms are usually afflicted by some problems in practical applications, including parameter dependence and the inability to generate clusters of arbitrary shapes. The majority of existing algorithms involve one or more input parameters, and their results are influenced by these parameters significantly. The k-means-like and spectral clustering algorithms require the number of clusters to be specified. While some methods have been proposed to determine the appropriate number of clusters, e.g., [11,12], this problem is still open in general. DBSCAN is able to calculate the number of clusters by itself, but it involves two density parameters $MinPts$ and Eps. The input of the AP algorithm is the preference of each data, which indicates the possibility of being a cluster center. In the DP algorithm, the density kernel usually involves parameters, which have a significant influence on the clustering results. The DSets algorithm uses the pairwise data similarity matrix as the input, and it does not involve any parameters explicitly. However, in the case that the data are represented as feature vectors, the similarity measure used to build the similarity matrix usually introduces parameters, which are found to impact on the clustering results [17]. With the mean shift algorithm, an inappropriately selected band width may cause unsatisfactory clustering results. Another problem is that many algorithms are only able to generate clusters of certain shapes. The examples include k-means, spectral clustering, AP and DSets, which tend to generate clusters of spherical shapes only.

A typical advantage of density based clustering algorithms over others lies in their ability to detect clusters of non-spherical shapes. These algorithms usually rely on local density to determine the cluster membership and the data with sufficiently large density can be included into clusters. Consequently, one data only needs to be similar to its nearest neighbors, and this makes it possible to generate clusters of arbitrary shapes. In contrast, in k-means-like algorithms each data in a cluster is required to be similar to the cluster center and the obtained clusters are usually spherical. The problem with density based clustering algorithms, including DBSCAN and others [1,2,16,29], is that they use a density threshold to determine the cluster border and one or more density parameters are involved. On one hand, it is not trivial to determine the appropriate density threshold without the prior information of the data distribution in clusters. On the other hand, in the case that different clusters have significantly different density, using a fixed density threshold to extract all the clusters may not be a good option.

In order to relieve the parameter dependence problem and generate clusters of arbitrary shapes, in this paper we present an approach based on the DSets and density based clustering algorithms. Our approach is motivated by the following observation. The DSets algorithm detects clusters sequentially, and this allows us to improve the clustering result significantly with a post-processing step after each cluster is obtained. Therefore we use the DSets algorithm to generate the initial clusters in the first step, and then obtain the final clusters with density based cluster growing. The advantage of this strategy is that density based cluster growing can be used to generate clusters of arbitrary shapes, and the involved parameters can be determined adaptively based on the initial clusters. As to the parameter dependence problem of the DSets algorithm, we use $s(i, j) = exp(-d(i, j)/\sigma)$ to evaluate the data similarity, where $d(i, j)$ is the Euclidean distance between data i and j, and eliminate the dependence on σ by histogram equalization of the pairwise data similarity matrix. Some works of this paper appeared in a preliminary version in [19].

The rest of this paper is organized as follows. In Section 2 we provide a brief introduction of the DSets and DP algorithms. The details of histogram equalization and density based cluster growing are then presented in Section 3. Extensive experiments are conducted to validate the effectiveness of histogram equalization transformation, cluster growing and the whole algorithm in Section 4. We conclude this paper in Section 5.

2 Background

Our approach is proposed on the basis of the DSets algorithm and density based clustering algorithms. In the following we present a brief introduction of the DSets algorithm and the DP algorithm, which is a special density based clustering algorithm.

2.1 DSets algorithm

Dominant set [24,26] is a graph based concept of a cluster. Informally, dominant set is the extension of the *clique* concept in graph theory to edge-weighted graphs. In other words, the data in a dominant set are highly similar to each other, and they are less similar to those outside the dominant set. With the pairwise data similarity matrix as the input, we extract a dominant set and regard it as the first cluster. Then we continue to extract the next cluster in the remaining unclustered data, and repeat this procedure until all the data are grouped into clusters. In this way the DSets algorithm generates the clusters sequentially and determines the number of clusters automatically.

The formal definition of dominant set is presented below. With n data to be clustered, we calculate their pairwise similarity matrix as $A = (a_{ij})$. We represent the data with an edge-weighted graph $G = (V, E, w)$, where V is the set of data, E denotes the relationship among the data, and w stands for the weight function. Evidently $w_{ij} = a_{ij}$ if $(i, j) \in E$ and $w_{ij} = 0$ otherwise. Since our aim is to do clustering and one data should not be similar to itself, G has no self-loops and $a_{ii} = 0$ for $i = 1, \cdots, n$. With S denoting the n data for clustering, D being a non-empty subset of S, $i \in D$ and $j \notin D$, a relationship between j and i is measured by

$$\phi_D(i,j) = a_{ij} - \frac{1}{|D|}\sum_{k\in D} a_{ik}, \tag{1}$$

where $|D|$ denotes the number of data in D. Then we define the following key variable

$$w_D(i) = \begin{cases} 1, & \text{if} \quad |D| = 1, \\ \sum\limits_{l \in D\setminus\{i\}} \phi_{D\setminus\{i\}}(l,i) w_{D\setminus\{i\}}(l), & \text{otherwise.} \end{cases} \tag{2}$$

The variable $w_D(i)$ measures the relationship of the average similarity between i and the data in D, with respect to the overall average similarity in D. Informally, $w_D(i) > 0$ indicates that the internal coherency of D is preserved after i is included into D. On the contrary, a negative $w_D(i)$ means that i is not very similar to the data in D and including i into D will destroy the internal coherency of D.

With $W(D) = \sum_{i\in D} w_D(i)$, a subset D such that $W(T) > 0$ for all non-empty $T \subseteq D$ is called a dominant set if

1. $w_D(i) > 0$, for all $i \in D$,
2. $w_{D \bigcup \{i\}}(i) < 0$, for all $i \notin D$.

This definition indicates that only the data with positive $w_D(i)$ are included into a dominant set and all those with negative $w_D(i)$ are excluded. Here we see that the importance of $w_D(i)$ lies in that it defines a non-parametric criterion of data admittance into a dominant set. In the case that the data are represented as a pairwise similarity matrix, this criterion enables us to extract dominant sets and accomplish the clustering process without the participation of parameters.

In [26] it is shown that game dynamics, e.g., replicator dynamics or the more efficient one proposed in [4], can be used to extract a dominant set. Then the clustering can be accomplished by extracting the dominant set (cluster) one by one. Some further works and applications of the dominant sets algorithm include [25,31,20,18,21].

2.2 DP algorithm

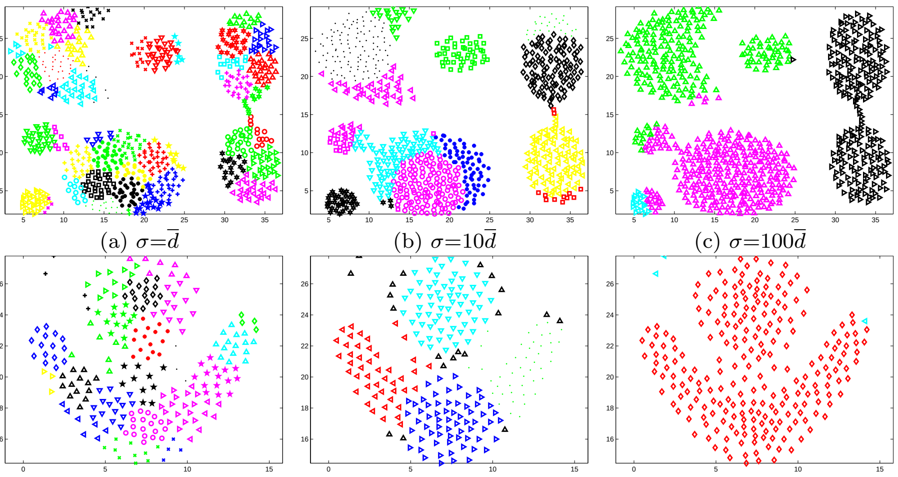
Density based clustering algorithms, e.g., DBSCAN, are usually based on density thresholds and the data with sufficiently large density are grouped into a cluster. In [27] density peak (DP) based clustering is presented from a different perspective. Observing that cluster centers are usually density peaks with larger density than the neighboring data, and that cluster centers are relatively far from each other, [27] designs variables ρ and δ to identify cluster centers and cluster members. Here ρ denotes the local density of a data, and δ is the distance between one data and its nearest neighbor with larger density. Obviously cluster centers are often far from each other and have large δ's, whereas non-center data are typically close to their neighbors with larger density and have small δ's. In summary, cluster centers are with both large ρ's and large δ's, whereas non-center data are usually with small δ's and possibly small ρ's. Based on this observation, [27] proposes to identify cluster centers with the $\rho - \delta$ decision graph, where cluster centers are isolated from non-center data.

After the cluster centers are identified, the non-center data are grouped into different clusters based on the density relationship between neighboring data. Specifically, [27] assumes that one non-center data has the same label as its nearest neighbor with larger density. Based on this assumption, the non-center data can be assigned labels efficiently by traversing the non-center data from large-density ones to small-density ones. While this assumption has no theoretical foundation, it is consistent with human intuition and is shown to work well in experiments.

From the above description we see that cluster center selection is the key of the DP algorithm, as it determines the number of clusters and influences the labels of all the non-center data. While it is true that cluster centers are with large ρ's and large δ's in the $\rho - \delta$ decision graph, it is not easy to determine cluster centers in many cases as there is no clear distinction between *large* and *small*. In [27] this problem is solved by manual selection in the decision graph, and some algorithms have been proposed in an attempt to determine the cluster centers automatically [33,34,9].

3 Proposed approach

Based on the DSets algorithm and density based clustering algorithms, the approach proposed in this paper consists of four major steps. Firstly, we use histogram equalization to transform the pairwise similarity matrix, so that the new similarity matrix is no longer influenced by the parameter σ. Then we feed the new similarity matrix to the DSets algorithm and obtain initial clusters. In the third step we use a density threshold based method to expand the initial clusters. Finally, a density peak based cluster growing method is used to improve the clustering results further. As the second step follows the standard DSets algorithm, we present the details of the other three steps in the following three subsections respectively.



(a) $\sigma=\overline{d}$ (b) $\sigma=10\overline{d}$ (c) $\sigma=100\overline{d}$

(d) $\sigma=\overline{d}$ (e) $\sigma=10\overline{d}$ (f) $\sigma=100\overline{d}$

Fig. 1 Aggregation and Flame Clustering results with different σ's. The top and bottom rows belong to Aggregation and Flame, respectively. It is shown that with both datasets, σ has a significant influence on the clustering results.

3.1 Histogram equalization transformation

The DSets algorithm uses only the pairwise similarity matrix as the input and does not involve any parameters in itself. However, if the data for clustering are represented as feature vectors, we need to evaluate the data similarity and build the similarity matrix, and parameters may be introduced in this process. With the commonly used $s(i,j) = exp(-d(i,j)/\sigma)$ as the similarity measure, the parameter σ influences the similarity values and then the clustering results. This problem can be illustrated with the following experiments conducted on the Aggregation dataset [15] and Flame dataset [14]. We use different values of σ's, including $\overline{d}$, $10\overline{d}$ and $100\overline{d}$, to build the similarity matrices and then do the clustering, where $\overline{d}$ is the average of pairwise Euclidean distances. The clustering results are reported in Figure 1, where both different symbols and different colors indicate different clusters.

Figure 1 shows that with the increase of σ, the obtained clusters become larger and the clustering results vary significantly. More examples of the influence from σ can be found in Section 4. This observation can be explained as follows. As the extension of the clique concept, dominant set requires the inside data to be highly similar to each other. With a small σ, many similarity values are small and the resulted clusters are also small. With the increase of σ, the similarity values rise and the obtained clusters become larger correspondingly. The influence of σ on clustering results indicates that in order to obtain good results, it is necessary to find out the appropriate σ. However, Figure 4 shows that there is no single σ which is appropriate for different datasets. As a result, we decide to solve the parameter dependence problem with a different approach.

In DSets clustering, different σ's result in different similarity matrices, which then lead to different clustering results. This observation implies that if we remove the influence of σ on the similarity matrix, we are able to eliminate the influence of σ on the clustering result. Therefore we study the influence of σ on the similarity matrix in the first step.

From $s(i,j) = exp(-d(i,j)/\sigma)$ we see that σ influences only the absolute magnitude of similarity values, and it cannot change the ordering of these similarity values. For example, if $d(i_1,j_1) > d(i_2,j_2)$, then we have $s(i_1,j_1) < s(i_2,j_2)$ for arbitrary positive σ's. This means that if we sort the similarity values in the ascending order according to their absolute magnitude, the ordering of these similarity values is fixed and not influenced by σ. Motivated by the observation that in histogram equalization the new data values are determined only by the ordering of original values, we use histogram equalization transformation of the similarity matrix to eliminate the influence of σ.

As an important image enhancement technique, histogram equalization is originally used to increase the overall intensity contrast in an image. By treating the similarity values as image pixels, we are able to transform similarity matrices with histogram equalization. We firstly quantize the similarity range into N bins and build a histogram of the similarity values h_k, $k = 1, 2, \cdots, N$. All the similarity values in the k-th bin are assigned the same new value as

$$r_k = \frac{1}{n^2} \sum_{j=1}^{k} h_j. \tag{3}$$

As n^2 is the total number of similarity values, the new value r_k equals the percentage of all the data in the k-th bin and in the bins with smaller values. If N is sufficiently large that each bin contains only data of the same value, then the new value of one data is determined by the percentage of the data with equal or smaller original values. In other words, after histogram equalization, the new values of data are determined only by the ordering of original values. This further means that the influence of σ on similarity values and on clustering results is eliminated completely. For ease of expression, in the sequel we use DSets-histeq to denote the DSets algorithm where the input similarity matrix has been transformed by histogram equalization.

Our approach is based in part on the DSets algorithm and requires the pairwise data similarity matrix as the input. If the data to be clustered are given in the form of feature vectors, we need to evaluate the data similarity and build the similarity matrix in the first place. In our approach we use $s(i,j) = exp(-d(i,j)/\sigma)$ to estimate the data similarity, where the parameter σ influences only absolute similarity values and cannot change the ordering of similarity values. Based on this observation, we use histogram equalization to eliminate the influence of σ on the similarity matrix and then on the clustering result. Since histogram equalization determines the new similarity values based on the ordering of the original similarity values, here we see that as long as the parameter does not change the ordering of similarity values, histogram

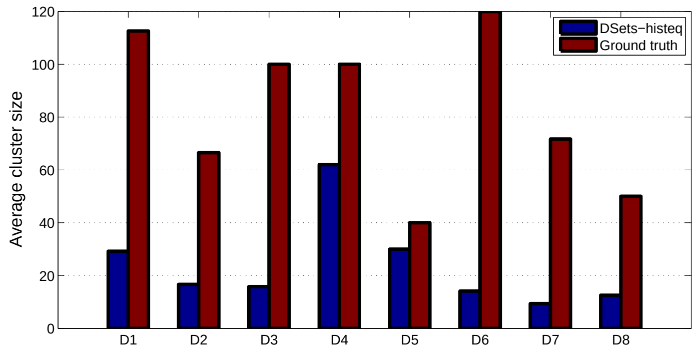


Fig. 2 Average cluster sizes from DSets-histeq and ground truth. The comparison shows that the clusters from DSets-histeq are usually smaller than the real ones.

equalization can be used to transform the similarity matrix generated with any similarity measures. The reason why $s(i, j) = exp(-d(i, j)/\sigma)$ is adopted in our approach is that it is one of the most commonly used similarity measures and works well in many cases. Whether or not it is the best choice does not influence the effectiveness of histogram equalization transformation in our approach.

3.2 Density threshold based growing

The dominant set definition imposes a high density constraint on the inside data. As a result, the DSets clustering results are sensitive to the absolute similarity values, and large clusters can only be obtained when the majority of similarity values are large. While histogram equalization transformation is able to eliminate the influence of σ on clustering results, it also changes the distribution of similarity values, and new similarity values are distributed rather evenly after the transformation. Since the majority of similarity values are small or medium, the obtained clusters tend to be small and the clustering results are not satisfactory. In fact, we show the average cluster sizes obtained with DSets-histeq and the comparison with the ground truth in Figure 2, and also the comparison of DSets-histeq results with the best of DSets results in Figure 3. The experiments are conducted on 8 datasets, and D1, D2, $\cdots$, D8 are used to denote the datasets in the order of Aggregation, Compound [35], Pathbased [5], D31 [32], R15 [32], Flame, Thyroid and Iris, where Thyroid and Iris are from the UCI repository. In Figure 3 the clustering results are evaluated with V-measure [28] with the homogeneity and completeness components weighted equally. These two figures confirm that DSets-histeq generates small clusters and the clustering results are usually not satisfactory.

Since DSets-histeq generates small clusters and results in unsatisfactory clustering results, one natural option to improve clustering results is to expand the clusters with some cluster growing method. Here we choose to expand the clusters based on density information in order to obtain clusters of arbitrary

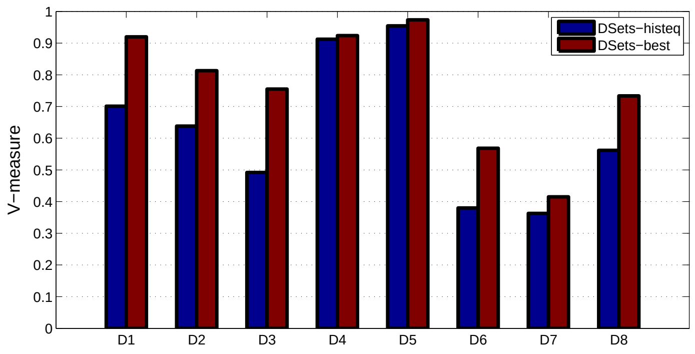


Fig. 3 DSets-histeq clustering results and the comparison with the best results of DSets. Here DSets-best is used to denote the peaks in Figure 4. The comparison shows that the results of DSets-histeq are much inferior to the best ones in general.

shapes. Our method is based on a very simple criterion, i.e., including the data with sufficiently large density into clusters, and whether the density is large enough is determined based on the initial clusters. Specifically, we use DSets-histeq to obtain a cluster, which is used as the initial cluster. We estimate the local density of each data in the initial cluster and select the minimum local density as the density threshold. Then for each outside data, if the local density is above the threshold, it will be included into the cluster. By repeating this process until all the data are grouped into clusters, we are able to accomplish the clustering process.

There are various ways to estimate local density. In our method, for each data i, we firstly find out its nearest neighbors in the cluster and obtain a subset D_{inn}. The local density is then estimated as the average of the pairwise similarities in D_{inn}. While the number p of nearest neighbors does have some influence on the clustering results, it can be limited to a small range reasonably. If p is too small, the estimated local density is sensitive to the noise. With a too large p, the obtained density may reflect the global density but not the local one. In our implementation we empirically set p as 10. It should be noted that in estimating the local density, the nearest neighbors must be selected in the cluster, even if the data itself is outside the cluster. This is to guarantee that the data are close enough to the cluster. In addition, we sort the outside data in decreasing order according to their average similarity to the cluster. The outside data closest to the cluster will be considered firstly.

3.3 Density peak based growing

In the first cluster growing step, we use a density threshold to determine if an outside data should be included into a cluster. With this criterion the minimum local density in the cluster does not decrease in the cluster growing process. While this criterion keeps the local density in the cluster at a relatively high level and guarantees the large internal similarity, we also notice that in many cases the data at the cluster border are of much smaller density than those in the central area. Evidently, these border data are likely to be excluded in the first cluster growing step. Therefore we propose to use a second cluster growing step to include the data of this kind.

In the DP algorithm, after the cluster centers are identified, the non-center data will be grouped into clusters based on the density relationship between neighboring data. Specifically, it is assumed that one data should be in the same cluster as its nearest neighbor with larger density. While this assumption has no theoretical foundation, it is reasonable by intuition and is shown to be effective in experiments [27]. This observation motivates us to expand the clusters further in the following way. Just as in the DP algorithm, we firstly need to calculate the local density of each data. For each outside data, we find its nearest neighbor with larger density. If this neighbor is in the cluster and their distance is smaller than a threshold, the data will be included. If this neighbor is not in the cluster, we continue to find out the nearest neighbor with larger density of this neighbor. Here the distance threshold is used to differentiate between the δ's of non-center data and of cluster centers. Specifically, we build a histogram of all the δ's and find the first bin which corresponds to a drop in the histogram and also categories at least 90 percent of the data as non-center data. Then the center value of this bin is determined as the threshold. Similar as in the first cluster growing step, the data closest to the cluster will be firstly considered.

While theoretically we can use only the second cluster growing step to expand the initial clusters, we find in experiments that using both steps generates better results. There are several possible reasons for this observation, including the complex data distribution, imperfect density kernels, and unsatisfactory distance threshold determining methods. In summary, at present we are still not able to guarantee that in a cluster only one data is identified as the cluster center, and all the non-center data in the cluster can be traced to this center based on the density relationship. Consequently, in this paper we stick to the two-step clustering growing procedure.

The whole procedure of our approach can be described with Algorithm 1.

4 Experiments

On the basis of the DSets algorithm, our algorithm introduces histogram equalization and cluster growing to eliminate the influence of σ and improve clustering results. In this part we firstly use experiments to validate the effect of these two parts separately. In addition, we study other possible approaches to achieve the independence of σ experimentally. Finally, a comparison is made between our algorithm and other algorithms with careful parameter tuning.

In the experiments of of Section 3 we use 8 datasets. Here we introduce 4 other datasets, namely Jain [22], A3 [23], Dim128 [13] and the Wdbc dataset from the UCI repository. The characteristics of these datasets are shown in

Algorithm 1 Our algorithm.

```
1: Calculate the pairwise similarity matrix;
2: Transform the similarity matrix by histogram equalization;
3: Assign the labels of all data to be 0;
4: $m \leftarrow 0$;
5: while There are data with labels equaling to 0 do
6:     $m \leftarrow m + 1$;
7:     Extract a cluster with the DSets algorithm;
8:     Expand the cluster with density threshold based growing algorithm;
9:     Expand the cluster with density peak based growing algorithm;
10:    for each data in the cluster do
11:        Set the label as $m$;
12:        Set the corresponding row in the similarity matrix as 0;
13:        Set the corresponding column in the similarity matrix as 0;
14:    end for
15: end while
```

Table 1 The datasets used in this paper.

	# of points	Data dimension	# of clusters
Aggregation	788	2	7
Compound	399	2	6
Pathbased	300	2	3
D31	3100	2	31
R15	600	2	15
Flame	240	2	2
Thyroid	215	5	3
Iris	150	4	3
Jain	373	2	2
Wdbc	569	32	2
A3	7500	2	50
Dim128	1024	128	16

Table 1. We will use only the 8 datasets in some experiments where the results are illustrated with figures in order to differentiate between different datasets clearly.

4.1 Histogram equalization

In this part we test if histogram equalization is really able to eliminate the influence of σ on the clustering result. We firstly apply the DSets algorithm to the datasets, and the clustering results with different σ's are reported in Figure 4. It is quite evident from Figure 4 that on all the 8 datasets the clustering results are influenced by σ's significantly.

It is shown in Section 3 that with a sufficiently large N, histogram equalization transformation of the similarity matrix is able to eliminate the influence of σ on the clustering result. In implementation, as the similarity values are continuous and the number of similarity values are usually large, it is typically impossible to build a histogram where each bin contains only data of

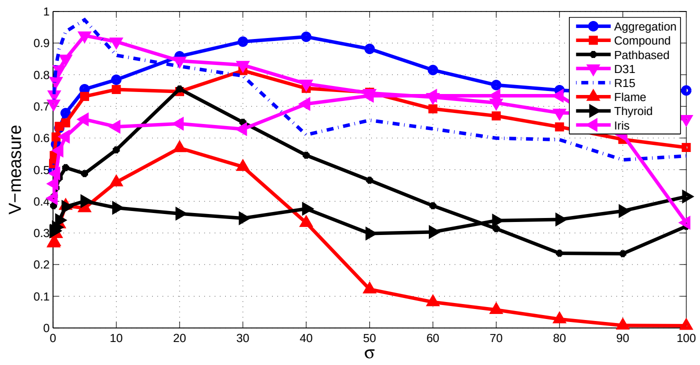


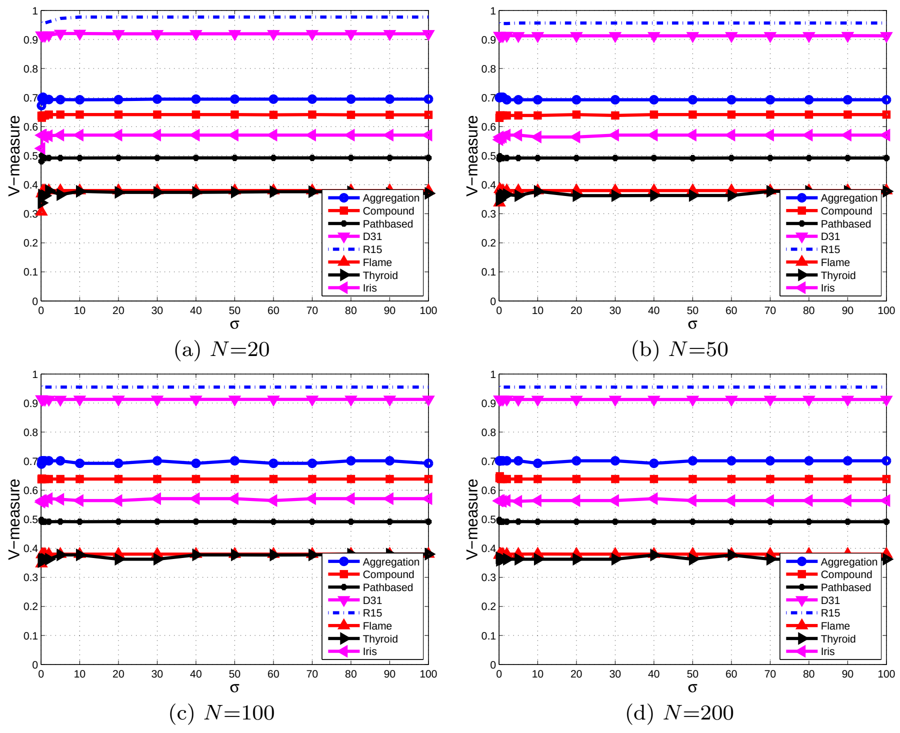
Fig. 4 DSets clustering results with different σ's. On all the eight datasets, the clustering results vary significantly with the change of σ.

identical value. With a small number N of bins, σ still exerts influence on the similarity matrix and then the clustering result. However, with the increase of N, this influence is reduced and can be ignored reasonably at some point. We investigate the impact of N on the clustering result as follows.

We use four values of N, namely 20, 50, 100 and 200, in histogram equalization transformation and feed the transformed similarity matrices to the DSets algorithm. The obtained clustering results with different σ's are reported in Figure 5.

From Figure 5 we observe that with all the four values of N, the parameter σ has rather little influence on the DSets-histeq clustering results. Comparatively, with the increase of N, the variance of clustering results with respect to σ is reduced. In addition, the clustering results are shown to be insensitive to N. Considering that the increase of N results in the increase of computation load, in our implementation we select $N = 100$, at which the influence of σ is already very small. Furthermore, we select $\sigma = \overline{d}$, although many other values can also be adopted.

As two cluster growing steps are introduced after histogram equalization, we then test if the whole algorithm is still independent of the parameter σ. The clustering results on the eight datasets are reported in Figure 6. While on some datasets we observe slight performance variations with the change of σ, generally we can conclude that the dependence on σ has been eliminated effectively. As to the relatively large variations at very small σ's on some datasets, our explanation is as follows. It is evident from $s(i,j) = exp(-d(i,j)/\sigma)$ that very small σ's lead to very small similarity values. In this case, a large amount of small similarity values are squeezed into small-value bins and are assigned the same new values after histogram equalization, resulting in an evident change of the similarity distribution. This further leads to the large variations of clustering results. In our work we select $\sigma = \overline{d}$, which is far from the dangerous small σ's. In summary, the histogram equalization transformation is shown to be effective in the whole algorithm.



(a) N=20 (b) N=50 (c) N=100 (d) N=200

Fig. 5 The influence of σ on DSets-histeq clustering results with different N's. Both σ and N have little influence on the clustering results. Notice that the major variations of clustering results occur when σ is relatively small.

4.2 With a fixed σ

In our algorithm we use histogram equalization to eliminate the influence of σ on clustering results. Another approach to avoid the dependence on σ is to use a fixed σ for different datasets. To compare these two approaches, we remove the histogram equalization step and use fixed σ's for different datasets, and call the new algorithm DSets-fixed. The comparison of our algorithm with DSets-fixed is reported in Figure 7, where the average clustering results on the eight datasets are used for comparison.

It can be seen from Figure 7 that with the change of σ, the clustering results of DSets-fixed vary significantly. In most cases DSets-fixed is outperformed by our algorithm evidently, and only at the peak it performs comparably to our algorithm. This observation can be explained as follows. With a small σ, the similarity values are small and we obtain very small clusters, which may not be enough to support reliable density estimation. As a result, the results after clustering growing are not satisfactory. In contrast, if σ is too large, the obtained clusters may be larger than the real ones, and in this case cluster growing only degrades the clustering results. Between the two extremes, an

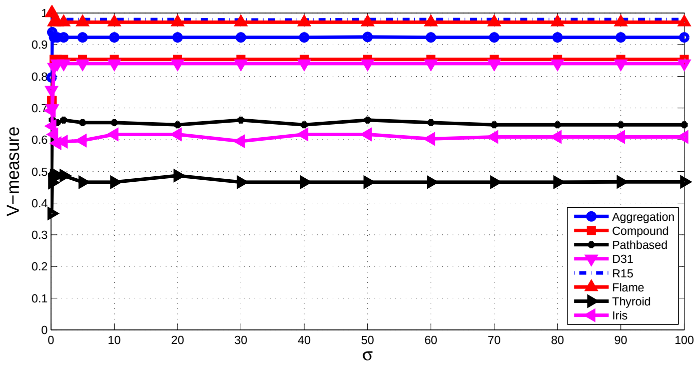


Fig. 6 The clustering results with our algorithm. Similar to Figure 5, the influence of σ on the clustering results is very small. This shows that the cluster growing steps do not change the independence of σ.

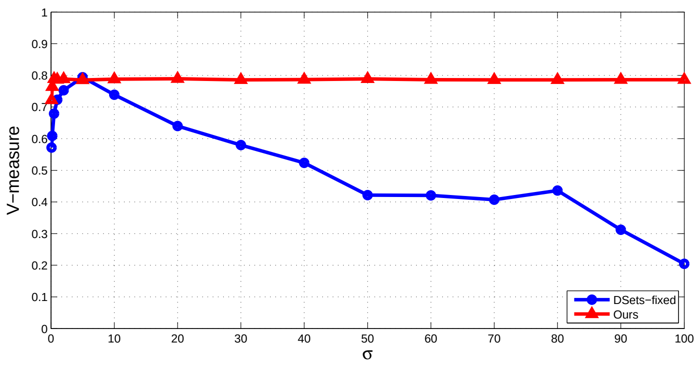


Fig. 7 The comparison of histogram equalization and fixed σ in our algorithm. It is shown that histogram equalization performs better than using a fixed σ in our algorithm.

appropriate σ may result in clusters which are not very small and not too large, and then lead to good clustering results. The problem is that different datasets usually have different appropriate σ's, and therefore it is hard to find a single σ appropriate for all the datasets. In contrast, with histogram equalization the similarity values are distributed in the range [0,1] rather evenly. The obtained clusters are usually small and not too small, and are suitable to be used in density based cluster growing. This guarantees that our algorithm is able to generate good results.

4.3 With non-parametric measures

In this paper we use $s(i,j) = exp(-d(i,j)/\sigma)$ to evaluate the data similarity and introduce the parameter σ. Noticing that there are some non-parametric similarity measures, e.g., cosine and histogram intersection kernel, it is also possible to use these similarity measures to avoid the influence of σ. In this

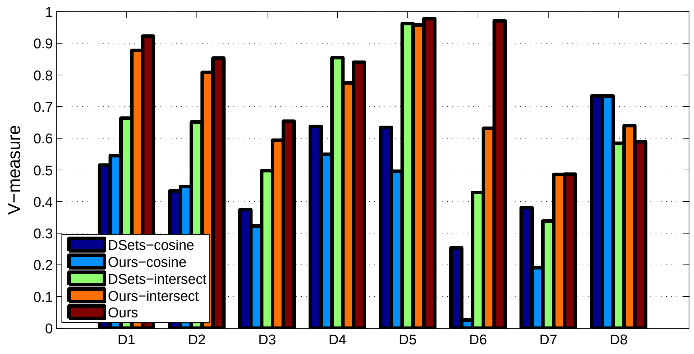


Fig. 8 The comparison of our algorithm with DSets-cosine, Ours-cosine, DSets-intersect and Ours-intersect.

part we use cosine and histogram intersection kernel to build the similarity matrix, and keep the other steps of our algorithm unchanged, and obtain two new algorithms called Ours-cosine and Ours-intersect. We also remove the two cluster growing steps and obtain two other algorithms DSets-cosine and DSets-intersect. The clustering results comparison of these four algorithms with our algorithm is shown in Figure 8.

We discuss the comparison shown in Figure 8 as follows. First, our algorithm performs the best or near-best on 7 out of the 8 datasets. In our algorithm histogram equalization transformation causes severe over-segmentation and the clusters from DSets-histeq are usually much smaller than the real ones, as shown in Figure 2. In this case, it is quite likely that each obtained cluster is only a subset of one single cluster. Consequently we can expand these clusters to increase the cluster sizes and improve the clustering results. Second, the Ours-cosine algorithm generates abysmal results on D6 and D7, and this is attributed to the cluster growing steps. Given a dataset, the DSets-cosine algorithm generates a fixed similarity matrix and then a fixed clustering result. The average cluster sizes from this algorithm is shown in Table 2, where it is evident that the obtained cluster sizes are close to the ground truth and even exceed the latter on some datasets. This implies a large possibility that each obtained cluster contains data from multiple clusters. In this case, cluster growing may degrade the clustering results significantly, as in the examples of D4, D5, D6 and D7. Comparatively, the clusters from DSets-intersect are much smaller than the real ones, and cluster growing improves the clustering results evidently on 6 out of the 8 datasets. Third, the Ours-cosine algorithm is the best-performing one on D8, and our algorithm is outperformed by both Ours-cosine and Ours-intersect on this dataset. As aforementioned, the cosine measure generates a fixed similarity matrix for a given dataset. The similarity matrix as the input may be good or bad for the DSets algorithm, and correspondingly we may obtain superior or terrible clustering result. As to the unsatisfactory result of our algorithm on D8, a possible reason is that the ini-

Table 2 Average cluster sizes with different algorithms.

	D1	D2	D3	D4	D5	D6	D7	D8
Ground truth	112.57	66.50	100.00	100.00	40.00	120.00	71.67	50.00
DSets-cosine	98.50	99.75	60.00	281.82	120.00	80.00	71.67	75.00
DSets-intersect	23.88	19.95	17.65	43.06	31.58	26.67	26.88	21.43
histeq	29.19	16.63	15.79	62.00	30.00	14.12	9.35	12.50

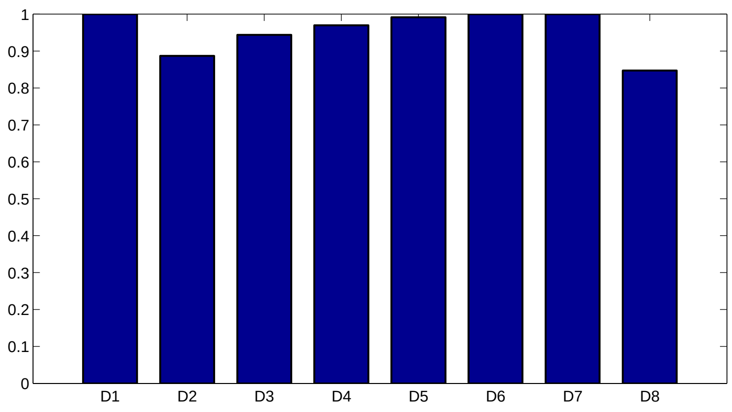


Fig. 9 The homoneity of initial clustering results.

tial clusters are too small, and cluster growing steps fail to solve this problem effectively. This leaves some space for improvement in the further work.

4.4 Cluster growing

In this paper we firstly generate initial clusters with DSets-histeq, and then improve the clustering results by expanding the initial clusters. The assumption behind this practice is that each initial cluster is small and contains data of one single class only. As discussed in Section 3, the definition of dominant set exerts a strong constraint on the high internal similarity of a dominant set, and histogram equalization transformation results in a large amount of small and medium similarity values. Consequently, the initial clusters tend to be small and consist of data of single clusters only. Here we use experiments to test if this assumption really holds. By considering only the data in the initial clusters, we evaluate the homogeneity of the clustering results, where the homogeneity is defined as in the V-measure. The results on the eight datasets are reported in Figure 9. It is quite evident that the homogeneity on all the datasets are rather large, indicating that the assumption holds to a large extent in our algorithm.

In order to check if the two cluster growing steps are really effective, we conduct experiments with three versions of our algorithm, namely without

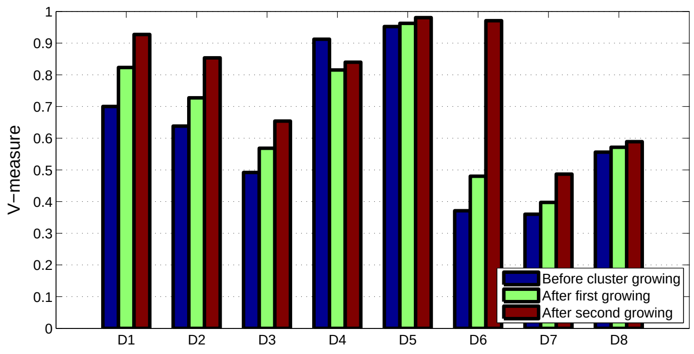


Fig. 10 The effect of two cluster growing steps in our algorithm.

cluster growing, with only the first cluster growing step, and with both cluster growing steps. The results of the three versions are shown in Figure 10.

Figure 10 indicates that on 7 out of the 8 datasets, the two cluster growing steps help improve the clustering results, validating the effectiveness of cluster growing in our algorithm. We discuss two interesting observations as follows. First, the first cluster growing step improves the clustering results on 7 datasets with the only exception on D4, namely the D31 dataset. In order to find out the reason, we show the clustering results obtained without cluster growing and with only the first growing step on this dataset in Figure 11. Evidently the clustering result without cluster growing is already quite good, and the only problem is that many points in the border areas are grouped incorrectly. After the first growing step, some clusters are merged due to our imperfect cluster growing method. This causes the decrease of clustering quality. After the second growing step, the cluster merging problem still exists, but many border points are grouped to the correct clusters, resulting in a slight increase of the clustering quality. Second, on the Flame dataset (D6) the first growing step improves the clustering result evidently, and then the second one increases the clustering quality significantly. The clustering results obtained with and without clustering growing on this dataset are shown in Figure 12. The first growing step is based on the density threshold and is only able to include nearby points with sufficiently large density into clusters. This leaves many small-density data unclustered and forces them to form new clusters, and consequently we observe that one cluster are partitioned into several small ones in Figure 12(c). However, the DP algorithm is designed to group neighboring small-density points into clusters, and therefore the second growing step generates large clusters and improves the clustering results significantly.

4.5 Comparison

After validation of the involved steps seperately, we compare the whole algorithm with some other algorithms, including the original DSets algorithm,

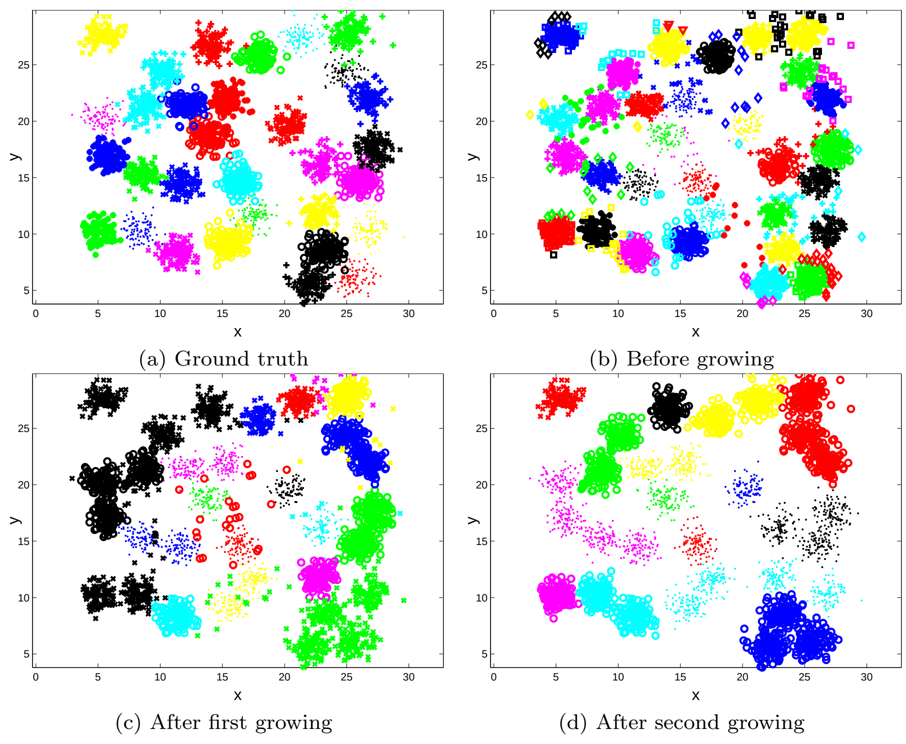
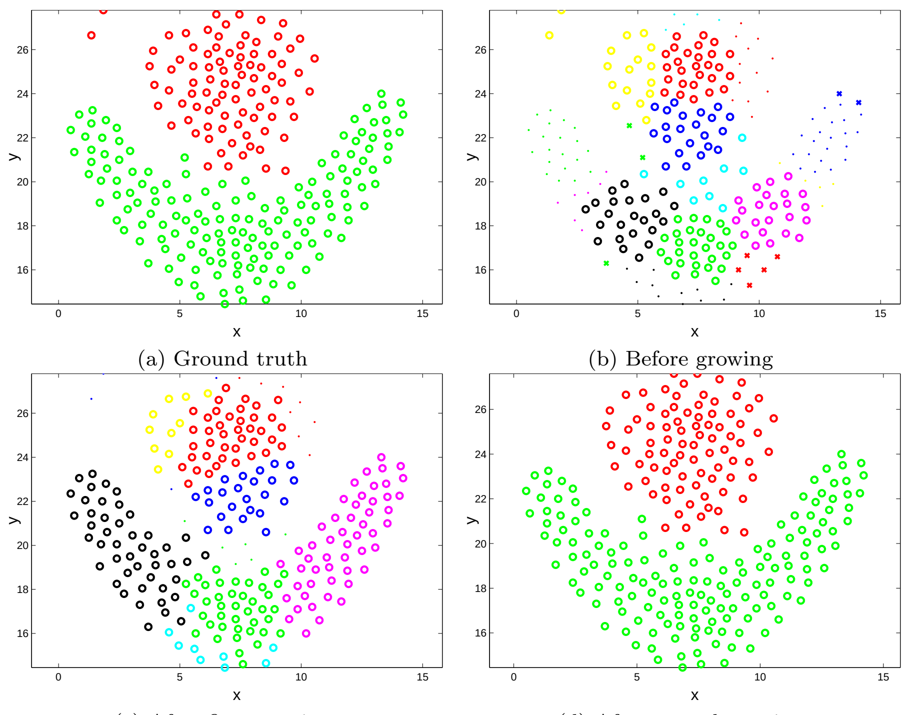


Fig. 11 The clustering result on the D31 dataset before and after cluster growing.

k-means, NCuts, DBSCAN, AP and DP. Except for the eight datasets used in previous experiments, here we add the Jain, Wdbc, A3 and DIM128 datasets. These four datasets increase the scope of dimension and dataset size of our experiments, and make our conclusions more convincing.

Noticing that all these algorithms for comparison involve one or more parameters as the input, we firstly introduce how the parameters are determined in our experiment.

1. With the k-means and NCuts algorithms, the required number of clusters is set as the ground truth.
2. In the two parameters required by DBSCAN, $MinPts$ is manually selected to be 3 from the testing values including 1, 2, $\cdots$, 10, and Eps is calculated based on $MinPts$ with the method presented in [8]. This parameter setting works for all the datasets except for Wdbc and DIM128, on which the generated Eps's are smaller than the minimum pairwise distance. Therefore we set $Eps = d_{min} + 0.2\lambda$, where d_{min} is the minimum distance to the $MinPts$ nearest neighbor, and $\lambda = d_{max} - d_{min}$ with d_{max} denoting the maximum distance to the $MinPts$ nearest neighbor.
3. In the DSets algorithm, we use $\sigma = 20\overline{d}$, which generates the best average results in the testing values including $0.1\overline{d}$, $0.2\overline{d}$, $0.5\overline{d}$, $\overline{d}$, $2\overline{d}$, $5\overline{d}$, $10\overline{d}$, $20\overline{d}$, $\cdots$, $100\overline{d}$.



(a) Ground truth
(b) Before growing
(c) After first growing
(d) After second growing

Fig. 12 The clustering results on the Flame dataset before and after cluster growing.

4. The AP algorithm requires the preference value p as the input, and the authors of [27] published the code to calculate the range $[p_0, p_1]$ of this parameter. In our experiment we use $p = p_0 + 9.2\xi$, which generates the best average results in all the testing values including $p_0 + \xi$, $p_0 + 2\xi$, $\cdots$, $p_0 + 9\xi$, $p_0 + 9.1\xi$, $p_0 + 9.2\xi$, $\cdots$, $p_0 + 9.9\xi$, where $\xi = (p_1 - p_0)/10$.
5. For the DP algorithm, we test the cutoff kernel and the Gaussian kernel and obtain two versions DP-c and DP-G. Both DP-c and DP-G need a percentage of neighboring data to calculate the cutoff distance d_c, and the percentage is recommended to be between 1% to 2% in [27]. In our experiment, this percentage is selected to be 1.1% for DP-c and 1.5% for DP-G, both of which generate the best average results in the testing values including 1%, 1.1%, $\cdots$, 2%. In these two DP algorithms, we use $\gamma = \rho\delta$ as the criterion to select cluster centers and the k data with the largest γ's are selected as cluster centers, where k is the number of clusters and assigned as the ground truth. In this way we avoid manual selection of cluster centers and the possible negative influence from imperfect cluster center selection methods.

From the above description we see that the algorithms for comparison have been assigned ground truth or manually selected parameters. In other words, their results shown in our experiments are approximately the best possible

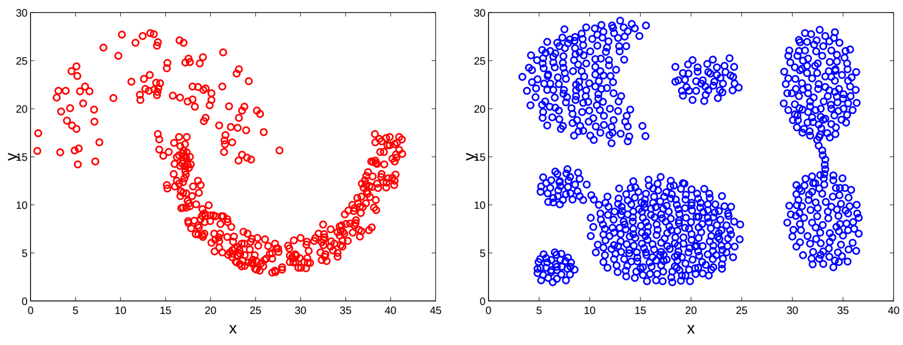
Table 3 The clustering results (V-measure) comparison of different algorithms.

	DSets	k-means	NCuts	DBSCAN	AP	DP-c	DP-G	Ours
Aggregation	0.86	0.85	0.76	0.92	0.81	**0.98**	0.88	0.92
Compound	0.75	0.72	0.62	**0.89**	0.80	0.79	0.76	0.85
Pathbased	**0.76**	0.55	0.50	0.63	0.54	0.55	0.54	0.65
D31	0.84	0.90	**0.96**	0.84	0.54	**0.96**	**0.96**	0.84
R15	0.83	0.84	**0.99**	0.87	0.71	0.98	**0.99**	0.98
Jain	0.38	0.36	0.33	**0.70**	0.40	0.57	0.60	0.49
Flame	0.57	0.46	0.44	0.83	0.54	**1.00**	0.41	0.97
Thyroid	0.36	0.43	0.32	0.19	0.24	0.08	0.14	**0.49**
Iris	0.65	0.74	0.74	0.72	0.79	0.65	**0.81**	0.59
Wdbc	0.38	0.46	**0.55**	0.05	0.39	0.17	0.34	0.37
A3	0.86	0.95	**0.99**	0.90	0.61	**0.99**	**0.99**	0.86
DIM128	**1.00**	0.95	**1.00**	0.75	0.39	**1.00**	**1.00**	0.98
mean	0.68	0.68	0.68	0.69	0.56	0.73	0.70	**0.75**

ones. As a result, if our algorithm generates comparable results with these algorithms, the effectiveness of our algorithm can be validated convincingly.

The comparison of these algorithms with ours is shown in Table 3. Obviously no algorithm performs the best on all the datasets, and this highlights the difficulty in developing an algorithm applicable to different types of datasets. Comparatively, DP-c, DP-G and NCuts generate the best results on the largest number (5) of datasets. Although our algorithm performs the best on only one dataset (Thyroid), its average result on all the 12 datasets is the best in all the 8 algorithms. In addition, the results of our algorithm are the best or near-best on 6 out of the 12 datasets. Considering that the algorithms for comparison benefit from ground truth or manually selected parameters, we believe these results demonstrate the effectiveness of our algorithm.

We discuss the results in Table 3 in a little more detail. The DBSCAN algorithm generates the best results on the Jain dataset and it performs the worst on Wdbc and nearly the worst on Thyroid. We explain this significant performance difference as follows. First, as illustrated in Figure 13(a), the Jain dataset is composed of two clusters whose shapes are far from being spherical. Evidently this dataset is difficult for k-means and NCuts as they tend to generate spherical clusters only. On the other hand, in the Jain dataset the two clusters are separated by a low-density area, and the data in each cluster are distributed rather evenly. This data structure is suitable for density based clustering algorithms, and it is not surprising that DBSCAN performs well on the Jain dataset. In fact, the other two density based algorithms, i.e., DP-G and DP-c are the second and third best-performing algorithms on the Jain dataset. Second, although the data in Thyroid and Wdbc are of high dimension (5 for Thyroid and 32 for Wdbc) and we cannot observe their data distribution visually in figures, we speculate that their data structure is difficult for density based algorithms. Our ground is that the three density based algorithms are exactly the three worst-performing ones on both Thyroid and Wdbc. This means that in these two datasets the data structure is rather complex and



(a) Jain (b) Aggregation

Fig. 13 The Jain and Aggregation datasets.

there is no evident density drop at the cluster borders. Another ground to support our speculation is that these two datasets are shown to be difficult for all the algorithms in our experiments, as the best V-measure values on Thyroid and Wdbc are only 0.49 and 0.55 respectively, much smaller than those on other datasets. In our opinion, the difficulty of these two datasets lies possibly in the existence of uninformative attributes. Both Thyroid and Wdbc are disease datasets and the data vectors are composed of measurable attributes. However, in collecting these attributes there is no knowledge as to which one is informative in indicating a disease. These attributes are just measured and then collected into a data vector. In this case, the existence of uninformative attributes may influence the distance calculation results and then impact on the clustering results. Possible solutions to this problem include new clustering algorithms, attribute selection methods and distance measures.

4.6 Computation efficiency

While histogram equalization is shown to be effective in removing the dependence on the parameter σ, it also causes small clusters. As a result, we have to use cluster growing steps to improve the clustering results. Both the histogram equalization transformation and cluster growing steps increase the computation load of the whole algorithm. In fact, Figure 14 shows the running time of the algorithms on different datasets, where the proposed approach is computationally more expensive than other algorithms except for AP. This is attributed in part to the large computation load of the DSets algorithm itself, and in part to histogram equalization and cluster growing steps. On the other hand, we have shown in Section 4.2 and Section 4.3 that neither using a fixed σ nor adopting non-parametric similarity measures is a good solution to achieve the independence of σ. Therefore in our future work we plan to make further study of the influence of σ in order to find out a more efficient approach to replace histogram equalization.

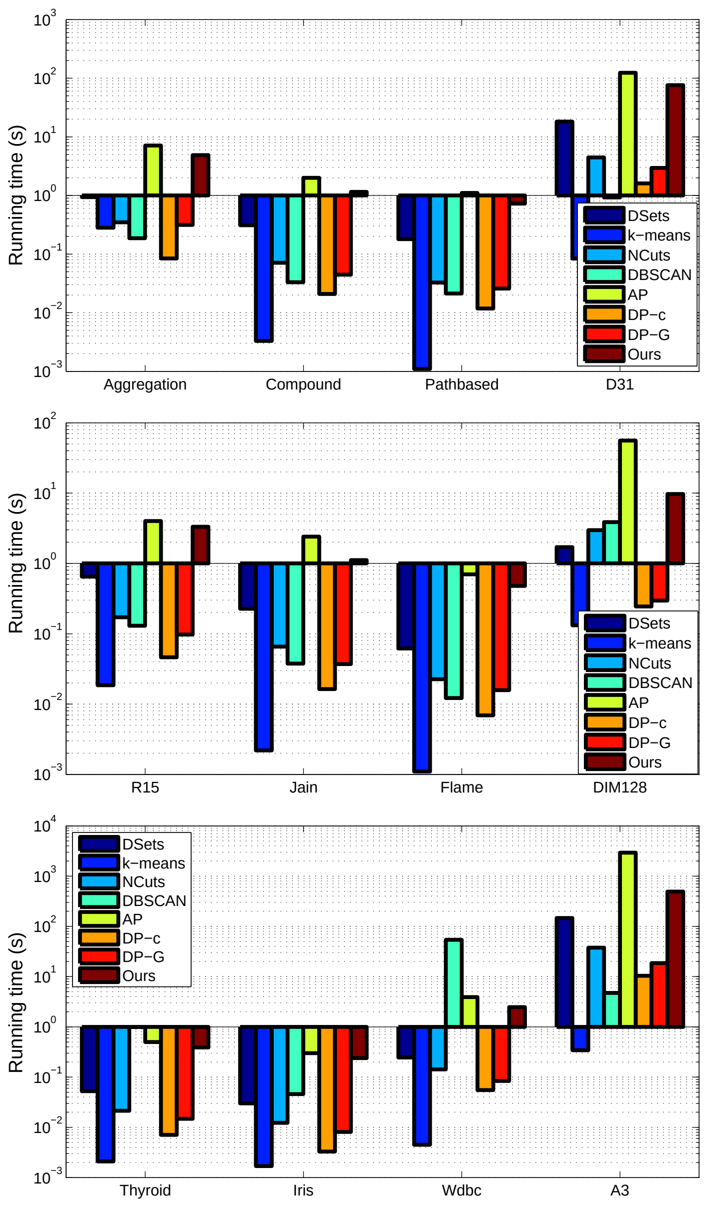


Fig. 14 Running time (in seconds) of different methods on the 12 datasets.

Some further observations from Figure 14 are as follows. First, the k-means algorithm is the most efficient one on 11 out of the 12 datasets, and the only exception is the Aggregation dataset, on which it is outperformed by DBSCAN and DP-c. It is reasonable that the k-means algorithm is very efficient due to its simplicity in implementation. The reason that DBSCAN and DP-c are more efficient on the Aggregation dataset, in our opinion, lies in the special data distribution in the dataset. As shown in Figure 13(b), the Aggregation dataset is composed of 7 clusters separated by evidently low-density areas. In each cluster, the data are distributed quite evenly and the density in all the clusters are similar. This is a perfect data structure for density based clustering algorithms, and consequently DBSCAN, DP-c and DP-G are able to accomplish the clustering with high accuracy efficiently. Second, AP is the most time-consuming algorithm on 10 out of the 12 datasets, with the exception on Thyroid and Wdbc, where DBSCAN is the most inefficient one. The large computation load of AP can be attributed to the clustering process of identifying cluster centers and members by passing affinity messages among data iteratively. As both Thyroid and Wdbc are composed of high-dimensional data and we cannot observe the data distribution visually, we haven't found out the straightforward reason of the unusually large running time of DBSCAN on Thyroid and Wdbc. However, we try to explain this observation by comparison. As aforementioned, the Aggregation dataset has a perfect data distribution for the DBSCAN algorithm, and consequently DBSCAN generates the near-best result with the least running time in the eight algorithms. In contrast, DBSCAN performs the worst on Wdbc and the third worst on Thyroid, and it is the most inefficient algorithm on both datasets. This comparison seems to imply that the data distribution of these two datasets is totally different from that of Aggregation. In other words, these two datasets are shown as the worst case for DBSCAN. As a result, DBSCAN generates terrible results with a large running time on Thyroid and Wdbc. In summary, in our experiments k-means is the most efficient algorithm, followed by DP-c and DP-G, and AP is the most time-consuming one, followed by our algorithm and DSets. DBSCAN and NCuts lie between these two extremes on the majority of the datasets.

5 Conclusions

In order to relieve the parameter dependence problem and generate clusters of arbitrary shapes, in this paper we present a density based cluster growing algorithm based on the dominant sets algorithm. By means of histogram equalization transformation, the parameter independent DSets-histeq algorithm is proposed to generate initial clusters, which are usually subsets of real clusters of not too small sizes. Cluster growing is then used to increase the cluster sizes and improve the clustering results, where the involved parameters are determined with the help of data in initial clusters. We present the density threshold based and density peak based growing steps to be used sequentially. Experiments indicate that the approach by histogram equalization outperforms the one based on fixed parameters and the one based on non-parametric similarity measures. It is also shown that both the two proposed clustering growing steps are able to improve the clustering results evidently. Comparisons with commonly used and state-of-the-art algorithms validate the effectiveness of the proposed approach.

References

1. Achtert, E., Bohm, C., Kroger, P.: Deli-clu: Boosting robustness, completeness, usability, and efficiency of hierarchical clustering by a closest pair ranking. In: International Conference on Knowledge Discovery and Data Mining, pp. 119–128 (2006)
2. Ankerst, M., Breunig, M.M., Kriegel, H.P., Sander, J.: Optics: Ordering points to identify the clustering structure. In: ACM SIGMOD International Conference on Management of Data, pp. 49–60 (1999)
3. Brendan, J.F., Delbert, D.: Clustering by passing messages between data points. Science **315**, 972–976 (2007)
4. Bulo, S.R., Pelillo, M., Bomze, I.M.: Graph-based quadratic optimization: A fast evolutionary approach. Computer Vision and Image Understanding **115**(7), 984–995 (2011)
5. Chang, H., Yeung, D.Y.: Robust path-based spectral clustering. Pattern Recognition **41**(1), 191–203 (2008)
6. Cheng, Y.: Mean shift, mode seeking, and clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence **17**(8), 790–799 (1995)
7. Comaniciu, D., Peter, M.: Mean shift: A robust approach toward feature space analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence **24**(5), 603–619 (2002)
8. Daszykowski, M., Walczak, B., Massart, D.L.: Looking for natural patterns in data: Part 1. density-based approach. Chemometrics and Intelligent Laboratory Systems **56**(2), 83–92 (2001)
9. Ding, J., Chen, Z., He, X., Zhan, Y.: Clustering by finding density peaks based on chebyshev's inequality. In: Chinese Control Conference, pp. 7169–7172 (2016)
10. Ester, M., Kriegel, H.P., Sander, J., Xu, X.W.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: International Conference on Knowledge Discovery and Data Mining, pp. 226–231 (1996)
11. Evanno, G., Regnaut, S., Goudet, J.: Detecting the number of clusters of individuals using the software structure: A simulation study. Molecular Ecology **14**(8), 2611–2620 (2005)
12. Fraley, C., Raftery, A.E.: How many clusters? which clustering method? answers via model-based cluster analysis. The Computer Journal **41**(8), 578–588 (1998)
13. Fränti, P., Virmajoki, O., Hautamäki, V.: Fast agglomerative clustering using a k-nearest neighbor graph. IEEE Transactions on Pattern Analysis and Machine Intelligence **28**(11), 1875–1881 (2006)
14. Fu, L., Medico, E.: Flame, a novel fuzzy clustering method for the analysis of DNA microarray data. BMC Bioinformatics **8**(1), 1–17 (2007)
15. Gionis, A., Mannila, H., Tsaparas, P.: Clustering aggregation. ACM Transactions on Knowledge Discovery from Data **1**(1), 1–30 (2007)
16. Hinnerberg, A., Keim, D.: An efficient approach to clustering large multimedia databases with noise. In: International Conference on Knowledge Discovery and Data Mining, pp. 58–65 (1998)
17. Hou, J., Gao, H., Li, X.: DSets-DBSCAN: A parameter-free clustering algorithm. IEEE Transactions on Image Processing **25**(7), 3182–3193 (2016)
18. Hou, J., Gao, H., Xia, Q., Qi, N.: Feature combination and the knn framework in object classification. IEEE Transactions on Neural Networks and Learning Systems **27**(6), 1368–1378 (2016)
19. Hou, J., Liu, W., E, X.: Density based clustering via dominant sets. In: IAPR-TC3 Workshop on Artificial Neural Networks in Pattern Recognition, pp. 80–91 (2016)
20. Hou, J., Liu, W., E, X., Cui, H.: Towards parameter-independent data clustering and image segmentation. Pattern Recognition **60**, 25–36 (2016)
21. Hou, J., Pelillo, M.: A simple feature combination method based on dominant sets. Pattern Recognition **46**(11), 3129–3139 (2013)

22. Jain, A.K., Law, M.H.C.: Data clustering: a user's dilemma. In: International Conference on Pattern Recognition and Machine Intelligence, pp. 1–10 (2005)
23. Kärkkäinen, I., Fränti, P.: Dynamic local search algorithm for the clustering problem. Research report A-2002-6, University of Joensuu (2002)
24. Pavan, M., Pelillo, M.: A graph-theoretic approach to clustering and segmentation. In: IEEE International Conference on Computer Vision and Pattern Recognition, pp. 145–152 (2003)
25. Pavan, M., Pelillo, M.: Efficient out-of-sample extension of dominant-set clusters. In: Advances in Neural Information Processing Systems, pp. 1057–1064 (2005)
26. Pavan, M., Pelillo, M.: Dominant sets and pairwise clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence **29**(1), 167–172 (2007)
27. Rodriguez, A., Laio, A.: Clustering by fast search and find of density peaks. Science **344**, 1492–1496 (2014)
28. Rosenberg, A., Hirschberg, J.: V-measure: a conditional entropy-based external cluster evaluation measure. In: Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 410–420 (2007)
29. Roy, S., Bhattacharyya, D.K.: An approach to find embedded clusters using density based techniques. LNCS **3816**, 523–535 (2005)
30. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence **22**(8), 167–172 (2000)
31. Torsello, A., Bulo, S.R., Pelillo, M.: Grouping with asymmetric affinities: a game-theoretic perspective. In: IEEE International Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 292–299 (2006)
32. Veenman, C.J., Reinders, M., Backer, E.: A maximum variance cluster algorithm. IEEE Transactions on Pattern Analysis and Machine Intelligence **24**(9), 1273–1280 (2002)
33. Wang, X.F., Xu, Y.: Fast clustering using adaptive density peak detection. Statistical Methods in Medical Research (2016)
34. Yang, N., Liu, Q., Li, Y., Xiao, L., Liu, X.: Star-scan: a stable clustering by statistically finding centers and noises. In: Asia-Pacific Web Conference on Web Technologies and Applications, pp. 456–467 (2016)
35. Zahn, C.T.: Graph-theoretical methods for detecting and describing gestalt clusters. IEEE Transactions on Computers **20**(1), 68–86 (1971)
36. Zhu, X., Loy, C.C., Gong, S.: Constructing robust affinity graphs for spectral clustering. In: IEEE International Conference on Computer Vision and Pattern Recognition, pp. 1450–1457 (2014)