Information Sciences 181 (2011) 3397–3410

Contents lists available at ScienceDirect

Information Sciences

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

Minimum spanning tree based split-and-merge: A hierarchical clustering method

Caiming Zhong [a,b,c], Duoqian Miao [a,*], Pasi Fränti [b]

[a] Department of Computer Science and Technology, Tongji University, Shanghai 201804, PR China
[b] Department of Computer Science, University of Eastern Finland, P.O. Box: 111, FIN-80101 Joensuu, Finland
[c] College of Science and Technology, Ningbo University, Ningbo 315211, PR China

ARTICLE INFO

Article history:
Received 6 December 2009
Received in revised form 4 April 2011
Accepted 6 April 2011
Available online 13 April 2011

ABSTRACT

Most clustering algorithms become ineffective when provided with unsuitable parameters or applied to datasets which are composed of clusters with diverse shapes, sizes, and densities. To alleviate these deficiencies, we propose a novel split-and-merge hierarchical clustering method in which a minimum spanning tree (MST) and an MST-based graph are employed to guide the splitting and merging process. In the splitting process, vertices with high degrees in the MST-based graph are selected as initial prototypes, and *K*-means is used to split the dataset. In the merging process, subgroup pairs are filtered and only neighboring pairs are considered for merge. The proposed method requires no parameter except the number of clusters. Experimental results demonstrate its effectiveness both on synthetic and real datasets.

© 2011 Elsevier Inc. All rights reserved.

1. Introduction

Clustering plays an important role in data mining, pattern recognition, and machine learning. It aims at grouping N data points into K clusters so that data points within the same cluster are similar, while data points in diverse clusters are different from each other. From the machine learning point of view, clustering is unsupervised learning as it classifies a dataset without any a priori knowledge. A large number of clustering algorithms [20,37,46] have been presented in the literature since *K*-means [32], one of the most popular clustering algorithms, was published. The algorithms can be grouped into hierarchical clustering [12,15,16,19,27], partitional clustering [2,6,17,24,32,35], density-based clustering [10,18], grid-based clustering [1,43], model-based clustering [11,33], and graph-based clustering [29,38,40,45,47,49]. Hierarchical and partitional clustering are the two most common groups [30].

Generally, a hierarchical clustering algorithm partitions a dataset into various clusters by an agglomerative or a divisive approach based on a dendrogram. Agglomerative clustering starts by considering each point as a cluster, and iteratively combines two most similar clusters in terms of an objective function. In contrast, divisive clustering starts with only one cluster including all data points. It iteratively selects a cluster and partitions it into two subclusters. The main advantage of hierarchical clustering is that it produces a nested tree of partitions and can therefore be more informative than non-hierarchical clustering. Furthermore, with the dendrogram, the optimal number of clusters can be determined. However, hierarchical clustering has a relatively high computational cost. Single linkage [39] and complete linkage [26] are two well-known examples of hierarchical clustering algorithms, and they take $O(N^2 \log N)$ time.

Partitional clustering splits a dataset at once using an objective function. *K*-means is one of the most popular examples of partitional clustering. It employs mean-squared-error as its objective function. Its main advantage is that it runs efficiently: its computational complexity is $O(NKId)$, where I is the number of iterations for convergence, and d is the dimensionality of

* Corresponding author. Tel.: +86 21 69589867.
E-mail address: miaoduoqian@163.com (D. Miao).

0020-0255/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2011.04.013

the dataset. Since K and d are usually far less than N, the algorithm runs in a linear time on low-dimensional data. However, there does not exist a universal objective function that can be used to discover all different intrinsic structures of datasets. Therefore, partitional clustering produces inaccurate results when the objective function used does not capture the intrinsic structure of the data. This is the reason why partitional clustering algorithms are incapable of dealing with clusters of arbitrary shapes, different sizes and densities.

Clustering algorithms that combine the advantages of hierarchical and partitional clustering have been proposed in the literature [5,23,25,28,30,31]. This kind of hybrid algorithms analyzes the dataset in two stages. In the first stage, the dataset is split into a number of subsets with a partitioning criterion. In the second stage, the produced subsets are merged in terms of a similarity measure. Different split and merge approaches have been designed in several hybrid algorithms. CSM [30] first applies K-means to partition the dataset into K' subsets, where K' is an input parameter. Afterwards, single linkage, which uses a dedicated cohesion function as the similarity measure, is utilized to iteratively merge the K' subsets until K subsets are achieved. In the split stage, as K-means may produce different partitions in different runs, the final results may be unstable.

CHAMELEON [23] is another example of a hybrid clustering algorithm. It constructs a K-nearest neighbor graph, and employs a graph cut scheme to partition the graph into K' subsets. Relative inter-connectivity and relative closeness are defined to merge the subsets. Liu et al. [31] proposed a multi-prototype clustering algorithm, which can also be considered as a hybrid method. The method uses a convergence mechanism, and repeatedly performs split and merge operations until the prototypes remain unchanged. However, many empirical parameters are involved. Kaukoranta et al. [25] proposed a split-and-merge algorithm, where the objective function is to minimize the mean squared error.

A minimum spanning tree (MST) is a useful graph structure, which has been employed to capture perceptual grouping [21]. Zahn defined several criteria of edge inconsistency for detecting clusters of different shapes [49]. However, for datasets consisting of differently shaped clusters, the method lacks an adaptive selection of the criteria. Xu et al. [47] proposed three MST-based algorithms: removing long MST-edges, a center-based iterative algorithm, and a representative-based global optimal algorithm. But for a specific dataset, users do not know which algorithm is suitable.

In this paper, we propose a minimum spanning tree based split-and-merge method (SAM). It works on numerical data and assumes that the graph can be calculated in a vector space. In the splitting stage, three iterations of MSTs are used to construct a neighborhood graph called 3-MST graph. The vertices with high degrees in the graph are selected as the initial prototypes, and K-means is then applied. In the merge stage, the neighboring subsets with respect to the MST are filtered out and considered for merge.

The rest of the paper is organized as follows: In Section 2, the proposed algorithm is described. The experimental results are presented in Section 3, and conclusions are drawn in Section 4.

2. The proposed method

2.1. Problem formulation

Given a set of data points $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_i, \ldots, \mathbf{x}_N\}$, where $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ij}, \ldots, x_{id})^T \in \Re^d$ is a feature vector, the goal of clustering is to partition the set X into K clusters: $C_1, C_2, \ldots, C_K$, where $C_i \neq \emptyset$, $C_i \cap C_j = \emptyset$, $X = C_1 \cup C_2 \cdots \cup C_K$, $i = 1{:}\ K$, $j = 1{:}\ K$, $i \neq j$. An undirected graph has been employed to represent a dataset for clustering [38,49]. Let $G(X) = (V, E)$ denote the undirected complete graph of X, the weights of the edges can be calculated with function $w : E \to \mathbb{R}$, where $V = X$, $E = \{(\mathbf{x}_i, \mathbf{x}_j) | \mathbf{x}_i, \mathbf{x}_j \in X, i \neq j\}$, and $w(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T (\mathbf{x}_i - \mathbf{x}_j)}$.

2.2. Overview of the split-and-merge algorithm

The algorithm consists of three main stages as illustrated in Fig. 1. First, an MST of the given dataset is constructed to guide the clustering process. In the splitting stage, K-means is applied to split the dataset into subsets, which are then adjusted according to the MST. In the merge stage, the final clusters are obtained by performing a carefully designed criterion-based merge that aims at maximizing intra-cluster similarity and minimizing inter-connectivity between the clusters.

2.3. The stage of constructing a 3-MST graph

2.3.1. Pruning leaves

A minimum spanning tree (MST) of graph $G(X)$ is a spanning tree T such that $W(T) = \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in T} w(\mathbf{x}_i, \mathbf{x}_j)$ is minimum. The leaves of an MST, called hairs in [49], are the vertices of degree 1. The leaves usually locate outside of kernels or skeletons of a dataset. For splitting two neighboring clusters, the data points in the neck [49] will have a negative effect on the clustering process. To alleviate the effect, we design a pruning step of removing the leaves so that the clusters are analyzed only on the essential data points. In [49], the skeleton of an MST was detected by repeatedly pruning until the number of removed hairs in two successive iterations remains the same. However, this may remove an entire cluster, and therefore, we conservatively perform only one pruning.

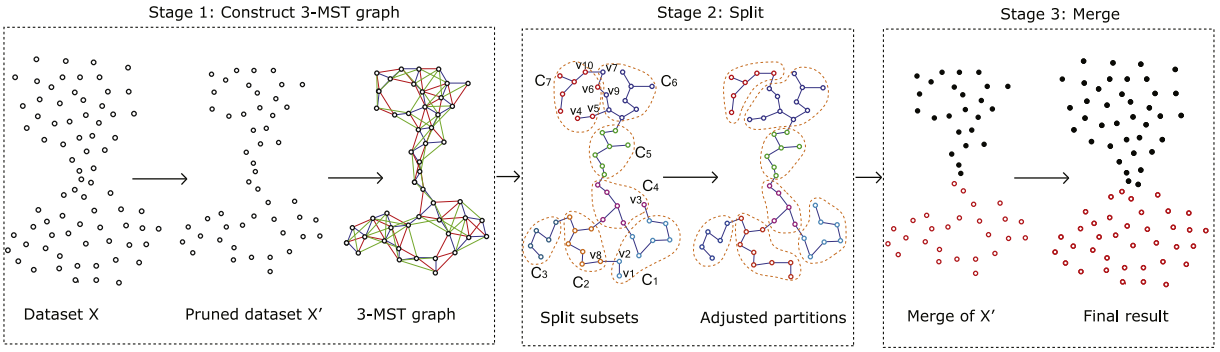

Fig. 1. The overview of split-and-merge. In Stage 1, the dataset X is pruned into X' according to the MST of X, and three iterations of MSTs of X' are computed and combined into a 3-MST graph. In Stage 2, X' is partitioned by K-means, where the initial prototypes are generated from the 3-MST graph. The partitions are then adjusted so that each partition is a subtree of the MST of X'. In Stage 3, the partitions are merged into the desired number of clusters and the pruned data points are distributed to the clusters.

Definition 1. Let X' be the pruned version of X as in

$$X' = X \setminus \{v_i | v_i \in V,\ degree(v_i) = 1\}, \tag{1}$$

where $degree(v_i)$ denotes the degree of vertex v_i in the MST of X.

2.3.2. Constructing a 3-MST graph

An MST describes the intrinsic skeleton of a dataset and accordingly can be used for clustering. In our proposed method, we use it to guide the splitting and merging processes. However, a single MST loses some neighborhood information that is crucial for splitting and merging. To overcome this drawback, we combine several MSTs and form a graph $G_{mst}(X,k)$ as follows:

Definition 2. Let $T_1 = f_{mst}(V,E)$ denote the MST of $G(X) = (V,E)$. The following iterations of an MST are defined as:

$$T_i = f_{mst}\left(V, E \setminus \cup_{j=1}^{i-1} T_j\right), \tag{2}$$

where f_{mst}: $(V,E) \to T$ is a function to compute an MST from graph $G(X) = (V,E)$, and $i \geqslant 2$.

In theory, the above definition of T_i is not rigorous because $E \setminus \cup_{j=1}^{i-1} T_j$ may produce isolated subgraph. For example, if there exists a vertex v in T_1 and the degree of v is $|V| - 1$, v will be isolated in $G(V, E \setminus T_1)$. Hence, the second MST (T_2) cannot be completed in terms of Definition 2. In practice, this is not a problem because the first MST T_1 is still connected and we can simply ignore it as a minor artefact, because it has no noticeably effect on the performance of the overall algorithm. However, for the sake of completeness, we solve this minor problem by always connecting such an isolated subgraph with an edge randomly selected from those connecting the isolated subgraph in T_1.

Let $G_{mst}(X,k)$ denote the k-MST graph, which is defined as a union of the k MSTs: $G_{mst}(X,k) = T_1 \cup T_2 \cup \cdots \cup T_k$. In this paper, we use $G_{mst}(X',k)$ to determine the initial prototypes in the split stage and to calculate the merge index of a neighboring partition pair in the merge stage. Here, k is set to 3 in terms of the following observation: 1 round of MST is not sufficient for the criterion-based merge but 3 iterations are. The number itself is a small constant and can be justified from computational point of view. Additional iterations do not add much to the quality, but only increase processing time. A further discussion concerning the selection of k can be found in Section 3.4.

2.4. Split stage

In the split stage, initial prototypes are selected as the nodes of highest degree in the graph $G_{mst}(X',k)$. K-means is then applied to the pruned dataset using these prototypes. The produced partitions are adjusted to keep the clusters connected with respect to the MST.

2.4.1. Application of K-means

The pruned dataset is first split by K-means in the original Euclidean space, where the number of partitions K' is set to $\sqrt{|X'|}$. This is done under the assumption that the number of clusters in a dataset is smaller than the square root of the number of patterns in the dataset [3,36]. If $\sqrt{|X'|} \leqslant K$, K' can be set to $K + \lambda(|X'| - K)$ to grantee that K' is greater than K, where $0 < \lambda < 1$. Since this is not a normal situation, we do not discuss the parameter λ in this paper. Moreover, if $|X'| \leqslant K$, the split-and-merge scheme will degenerate into a traditional agglomerative clustering.

However, to determine the K' initial prototypes is a tough problem, and a random selection would give an unstable splitting result. For example, the method proposed in [30] uses K-means with randomly selected prototypes in its split stage, and the final clustering results are not unique. We therefore utilize the MST-based graph $G_{mst}(X',3)$ to avoid this problem.

Definition 3. Let v_i be the ith prototype from $G_{mst}(X',3)=(V,E)$, $V_{i-1}=\{v_1,v_2,\ldots,v_{i-1}\}$, $E_{i-1}=\{(\mathbf{x}_i,\mathbf{x}_j)|(\mathbf{x}_i,\mathbf{x}_j)\in E\wedge(\mathbf{x}_i=v\vee\mathbf{x}_j=v)\wedge v\in V_{i-1}\}$, v_i is generated as:

$$v_i=\arg\max_{v\in V\setminus V_{i-1}} Car\big(\{(\mathbf{x}_i,\mathbf{x}_j)|(\mathbf{x}_i,\mathbf{x}_j)\in(E\setminus E_{i-1})\wedge(\mathbf{x}_i=v\vee\mathbf{x}_j=v)\}\big), \tag{3}$$

where $Car(S)$ denotes the cardinality of set S.

The above definition determines the vertex with the maximum degree as a prototype. It is a recursive definition, when we determine the ith prototype v_i, the edges connected to the existing $i-1$ prototypes will not be considered for counting the degrees of vertices. However, there may exist two or more vertices in $G_{mst}(X',3)$ simultaneously having the maximum degree. In this case, the vertex with the minimum sum of weights of its edges is to be selected.

After the K' initial prototypes have been determined, K-means is applied. The seven subsets $(C_1,\ldots,C_7)$ in stage 2 of Fig. 1 are the partitions on T_1 produced by K-means.

2.4.2. Adjusting partitions

In traditional MST-based clustering methods such as [47,49], each partition is a subtree of the MST rather than a forest. This is a reasonable observation, because data points in the same cluster are usually in the same branch of the MST. However, the partitions in stage 2 of Fig. 1 may not coincide with the observation. For example, the four subsets (C_1,C_4,C_6,C_7) are forests but not trees.

Furthermore, the partitions in stage 2 of Fig. 1 reduce the ability of the MST to guide the merging process. This is because only neighboring partitions will be considered to be merged, and the neighboring partitions can be easily determined by the MST. However, if the partitions are forests, the MST would lose the ability.

Therefore, a process of redistributing the vertices is designed to transform a forest into a tree. The process is described as follows. Suppose that T_1 of X' is partitioned into K' forests, which are denoted as $F_1,F_2,\ldots,F_{K'}$. For redistributing the vertices, the main tree in each forest is defined.

Definition 4. Let $F_i=\{t_1,t_2,\ldots,t_j,\ldots,t_n\}$, and each t_j being a tree. The main tree of F_i is defined as:

$$Maintree(F_i)=\arg\max_{t_j\in F_i} Car(t_j), \tag{4}$$

where $Car(t_j)$ denotes the edge number of the tree t_j.

The vertices which are not included in any main tree will be re-allocated so that each subset is a subtree of T_1 of X'. Suppose $F_1,\ldots,F_7$ are the forests of the subsets $C_1,\ldots,C_7$ in Fig. 1, respectively. To adjust the seven subsets into seven subtrees, vertices $v1$ and $v2$ can be re-allocated to $Maintree(F_2)$, $v3$ to $Maintree(F_1)$, $v4$, $v5$, $v6$ to $Maintree(F_6)$, and $v7$ to $Maintree(F_7)$. The re-allocation process is described as follows.

Let SV denote the set of vertices that are not in any main tree, ST the set of the main trees. The redistribution operation is defined as: for an edge e_{ab} from T_1 of X', if $a\in SV$, and $\exists T(T\in ST\wedge b\in T)$, then the vertex a is redistributed to T. For example, $v2$ is re-allocated to $Maintree(F_2)$, since $e_{v2v8}\in T_1$, $v2\in SV$, and $v8\in Maintree(F_2)$. We iteratively perform this operation until all of the vertices in SV are re-allocated to the main trees.

The above operation may produce non-unique redistributions. Take $v6$ and $v7$ for example, if e_{v6v9} and e_{v7v6} are considered before e_{v7v10}, then $v6$ and $v7$ will be redistributed to $Maintree(F_6)$. Similarly, $v6$ and $v7$ may be redistributed to $Maintree(F_7)$, or $v6$ to $Maintree(F_6)$ and $v7$ to $Maintree(F_7)$. However, the non-unique redistribution does not noticeably affect the final clustering result for the two reasons: one is that the subtrees in a subset are usually smaller than the main tree of the subset, the other one is that the non-uniqueness most often happens inside rather than between the expected clusters.

2.5. Merge stage

After X' has been split into K' subgroups, the merge stage is performed to obtain the final clusters. In the merging process, the crucial problem is to determine which pairs of subgroups should be merged. By brute force, there are $K'\times(K'-1)/2$ candidate pairs for merge. In this paper, a stepwise refinement method is taken to address the problem. At the beginning, T_1 is employed to filter out the pairs. On the basis of MST-based clustering [47,49], unconnected subtrees cannot be merged. For example, the subgroups C_3 and C_7 in Fig. 1 will not be considered as a pair for merge. Consequently, only the neighboring pairs with respect to T_1 are the candidates.

2.5.1. Neighboring pairs

Definition 5. Let *Pairs* be the set of neighboring pairs from X' as in:

$$Pairs=\{(C_i,C_j)|\exists(\mathbf{x}_p,\mathbf{x}_q)\in T_1,\ (\mathbf{x}_p\in C_i\wedge\mathbf{x}_q\in C_j)\vee(\mathbf{x}_p\in C_j\wedge\mathbf{x}_q\in C_i)\}, \tag{5}$$

where $i\neq j$.

For any two subgroups, if there exists an edge in T_1 connecting them, then the two subgroups belong to *Pairs*.

For selecting the best pair to be merged, we define an *inter-connectivity index* and an *intra-similarity index* to rank the pairs.

2.5.2. Inter-connectivity index

Definition 6. Suppose $(C_i, C_j) \in Pairs$. In $G_{mst}(X', 3)$, let $E_{inter}(C_i, C_j)$ be the set of edges across the partitions C_i and C_j:

$$E_{inter}(C_i, C_j) = \{(\mathbf{x}_p, \mathbf{x}_q) | ((\mathbf{x}_p \in C_i \wedge \mathbf{x}_q \in C_j) \vee (\mathbf{x}_p \in C_j \wedge \mathbf{x}_q \in C_i))\}. \quad (6)$$

In Fig. 2, $E_{inter}(C_6, C_7) = \{(a,g), (a,h), (b,g), (c,i), (d,i), (d,j), (e,i), (e,j), (e,l), (e,k)\}$, i.e. the edges crossing the dotted curve.

Definition 7. Suppose that $(C_i, C_j) \in Pairs$. In $G_{mst}(X', 3)$, set $V_{i,j} = \{\mathbf{x}_p | (\mathbf{x}_p, \mathbf{x}_q) \in E_{inter}(C_i, C_j) \wedge \mathbf{x}_p \in C_i\}$, $E_{i,j}$ is a set of edges within C_i where at least one endpoint is in $V_{i,j}$:

$$E_{i,j} = \{(\mathbf{x}_p, \mathbf{x}_q) | \mathbf{x}_p, \mathbf{x}_q \in C_i \wedge (\mathbf{x}_p \in V_{i,j} \vee \mathbf{x}_q \in V_{i,j})\}. \quad (7)$$

In Fig. 2, for example, $V_{6,7} = \{g,h,i,j,k,l\}$, $V_{7,6} = \{a,b,c,d,e\}$, $E_{6,7}$ includes all the internal edges of C_6 (i.e. the edges whose both endpoints are from C_6) except $\{(n,m), (n,o), (n,p), (m,o), (m,p), (m,q), (p,o), (p,q)\}$, and $E_{7,6}$ includes all the internal edges of C_7. Actually, $E_{i,j}$ is defined as the edges in the region of C_i that is close to C_j.

Connection span is then defined with respect to $G_{mst}(X', 3)$ as a factor of measuring the similarity of two clusters based on the width of their connection. The intuition behind this index is that it estimates the size of the common border of the two clusters. The larger the common border is, the higher the priority for merging these clusters will be. It depends only on the distances in vector space and makes no assumption on the dimensionality.

Definition 8. Suppose that $(C_i, C_j) \in Pairs$. In $G_{mst}(X', 3)$, the connection span of C_i with respect to C_j is:

$$ConnSpan_{i,j} = \max_{\mathbf{x}_p, \mathbf{x}_q \in V_{i,j}} w(\mathbf{x}_p, \mathbf{x}_q). \quad (8)$$

In Fig. 3, the connection span of C_6 and C_7 is marked by the dotted edges (g,k) and (a,e), respectively.

Definition 9. Suppose that $(C_i, C_j) \in Pairs$. In $G_{mst}(X', 3)$, the inter-connectivity (IC) of C_i and C_j is defined as:

$$IC(C_i, C_j) = \frac{|E_{inter}(C_i, C_j)|}{\min(|C_i|, |C_j|)} \times \frac{\min(Avg(E_{i,j}), Avg(E_{j,i}))}{Avg(E_{inter}(C_i, C_j))} \times \max\left(ConnSpan_{i,j}, ConnSpan_{j,i}\right), \quad (9)$$

where $Avg(E)$ denotes the average weight of an edge set E.

From Eq. (9), the inter-connectivity index IC is a composite of three factors. The factor $|E_{inter}(C_i, C_j)|/\min(|C_i|, |C_j|)$ describes that the more edges straddling a pair of clusters, the stronger is the connective strength between the two clusters. In the second factor, $\min(Avg(E_{i,j}), Avg(E_{j,i}))$ is the minimum of the average weights of the edges that are in the two close regions from C_i and C_j, respectively. $Avg(E_{inter}(C_i, C_j))$ is the average weight of the edges straddling C_i and C_j. This factor reflects that the ratio of the average weight of the straddling edges to the minimum average weight in the two close regions is inversely proportional to the connectivity of the two clusters. In fact, this is based on the observation that if the density between the two clusters is low compared with those of the close regions, the two clusters have a small probability to be merged. The third factor $\max(ConnSpan_{i,j}, ConnSpan_{j,i})$ indicates that a pair of clusters with large connection span has strong connectivity.

2.5.3. Intra-similarity index

For describing the intra-similarity of pairs of clusters, a strategy inspired by Karypis et al. [23] is employed. Each cluster of a neighboring pair is bisected in terms of T_1, and the corresponding inter-edges with respect to $G_{mst}(X', 3)$ are used to evaluate the intra-similarity between the two clusters. The process is described as follows.

Suppose C_i is the cluster to be bisected, and T_{i1} is the subtree of T_1 restricted to nodes and edges of C_i. The bisecting edge $e_{bisect} \in T_{i1}$ can be determined as:

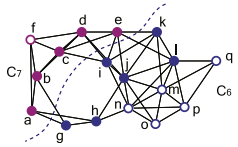

Fig. 2. Illustration of the inter edges between subgroups C_6 and C_7.

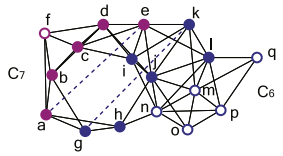

Fig. 3. Illustration of *connSpan* between subgroups C_6 and C_7.

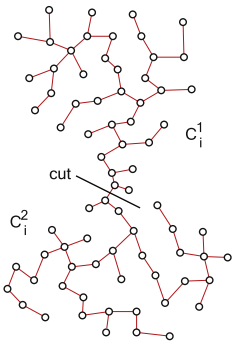

Fig. 4. Illustration of the bisection process of a cluster C_i. The data set is bisected into C_i^1 and C_i^2 by the cut so that the difference of cardinalities of C_i^1 and C_i^2 is minimal.

$$e_{bisect} = \arg\min_{e_j \in T_{i1}} \left| Car(t_{j1}) - Car(t_{j2}) \right|, \tag{10}$$

where t_{j1} and t_{j2} are two subtrees of T_{i1} generated by removing e_j from T_{i1}, and $Car(t)$ denotes the number of edges in a tree t. An example of the bisection is shown in Fig. 4. Correspondingly, C_i is bisected into two subsets: C_i^1 and C_i^2. The inter-edge set $E_{inter}(C_i^1, C_i^2)$ with respect to $G_{mst}(C_i, 3)$ is obtained by Definition 6. The intra-similarity of C_i and C_j is then defined as follows:

Definition 10. Suppose that $(C_i, C_j) \in Pairs$, C_i is bisected into C_i^1 and C_i^2, C_j is bisected into C_j^1 and C_j^2, the *intra-similarity* (*IS*) of the pair (C_i, C_j) is defined as:

$$IS(C_i, C_j) = \frac{1}{r_1 \times r_2} \times \frac{\sqrt{Avg\left(E_{inter}\left(C_i^1, C_i^2\right)\right) \times Avg\left(E_{inter}\left(C_j^1, C_j^2\right)\right)}}{Avg\left(E_{inter}\left(C_i^1, C_i^2\right)\right) + Avg\left(E_{inter}\left(C_j^1, C_j^2\right)\right)}, \tag{11}$$

where r_1 and r_2 are the numbers of edges of $E_{inter}(C_i^1, C_i^2)$ and $E_{inter}(C_j^1, C_j^2)$, respectively, and $Avg(E)$ denotes the average weight of an edge set E.

Eq. (11) implies that the intra-similarity between the pair (C_i, C_j) is high when the two averages $Avg(E_{inter}(C_i^1, C_i^2))$ and $Avg(E_{inter}(C_j^1, C_j^2))$ are close to each other. For the two numbers r_1 and r_2, unbalanced and small sizes indicate that the two clusters are more likely to be merged.

Taking into account the inter-connectivity and the intra-similarity as a whole, we define the overall merge index as:

$$R(C_i, C_j) = IC(C_i, C_j) \times IS(C_i, C_j). \tag{12}$$

The neighboring pair with the highest $R()$ value is merged first. After one neighboring pair is merged, *Pairs* and corresponding $R()$ values are updated. When K partitions are achieved, the merge process stops. Because a pruned leaf $\in X$ is connected to exact one vertex $\mathbf{x}_i \in X'$ in the MST of X, it is assigned the same label as $\mathbf{x}_i$.

Finally, the proposed algorithm is described as follows:

Algorithm 1: Split-and-merge (SAM)

Input: Dataset X, number of clusters K
Output: K clusters $C_1, C_2, \ldots, C_K$

1. Construct 3-MST graph
   - 1.1 Construct an MST on X
   - 1.2 Produce X' by pruning the leaves of the MST
   - 1.3 Create three MSTs on X': T_1, T_2 and T_3
   - 1.4 Compute 3-MST graph based on X': $G_{mst}(X', 3) \leftarrow T_1 \cup T_2 \cup T_3$
2. Split the pruned dataset X' into clusters
   - 2.1 Select K' highest degree nodes from $G_{mst}(X', 3)$ as initial prototypes
   - 2.2 Apply K-means with the prototypes to produce K' partitions
   - 2.3 For each of the partitions, find its main tree in T_1
   - 2.4 For each of the subtrees, repeatedly combine it with another subtree until it belongs to a main tree
3. Merge the partitions into final clusters
   - 3.1 Generate the set of neighboring partition pairs *Pairs*
   - 3.2 For each pair $(C_i, C_j) \in Pairs$, calculate the merge criterion $R(C_i, C_j)$
   - 3.3 Repeatedly merge the pair with maximum $R()$-value until K clusters have been obtained
   - 3.4 Add the pruned leaves to the clustering using T_1

2.6. Computational complexity

To construct an MST, if Fibonacci heaps are employed in Prim's algorithm, the running time is $O(|E| + |V|\log|V|)$ [9]. In this paper, an MST is computed from a complete graph, $|E| = O(|V|^2) = O(N^2)$, and therefore, the complexity of constructing an MST is $O(N^2)$.

In Step 1.1, the MST is constructed in $O(N^2)$ time, and in Step 1.2, the complexity of pruning the leaves from the MST is $O(N)$. In Step 1.3, to generate the three MSTs of X' takes $O(N^2)$ time. $G_{mst}(X', 3)$ is obtained in $O(N)$ time in Step 1.4. In Step 2.1, the initial prototypes are determined in $O(N)$, the time complexity of K-means on X' is $O(K'NdI)$, where d is the dimensionality of X', and I is the number of iterations. Since $K' \leqslant \sqrt{N}$, the total complexity of Step 2.1 is $O(N^{3/2}dI)$. The main trees can be determined in $O(N)$ in Step 2.2, and re-allocation can also be achieved in $O(N)$ in Step 2.3. Step 3.1 takes $O(N)$ to generate *Pairs*, and Step 3.2 takes $O(N)$ to compute the merge index $R()$ for every pair in *Pairs*. As the merging of a pair can be achieved in constant time, the maximum complexity of updating the merge index $R()$ is $O(N)$, and the number of iterations of Step 3.3 is $O(\sqrt{N})$, the worst case complexity of this Step is therefore $O(N^{3/2})$. Step 3.4 can be processed in constant time.

To sum up, the computational complexity of the proposed method (SAM) is $O(N^2)$, which is dominated by the construction of the 3-MST graph. If also the factor of dimensionality d is considered, the exact complexity would be $O(N^2d)$.

3. Experimental results

The clustering performance of the proposed method is evaluated on six synthetic and four real datasets. The first four synthetic datasets DS1–DS4 are taken from the literature [22,4,14,13], and the next two DS5, DS6 are from [42]. These datasets are illustrated in Fig. 5. The four real world instances are taken from the UCI datasets [50], including IRIS, WINE, Breast Cancer Wisconsin (WBC), and Breast Cancer Wisconsin Diagnostic (WDBC). The descriptions of these datasets are shown in Table 1.

The proposed method SAM is compared to the following six clustering algorithms:

1. K-means [32].
2. DBScan [10].
3. Single linkage [39].
4. Spectral clustering [38].
5. CSM [30].
6. CHAMELEON [23].

The first four algorithms are traditional clustering methods, whereas the next two are hybrid ones. In addition, three variants of SAM are performed to demonstrate the importance of the various steps and design choices of the algorithm:

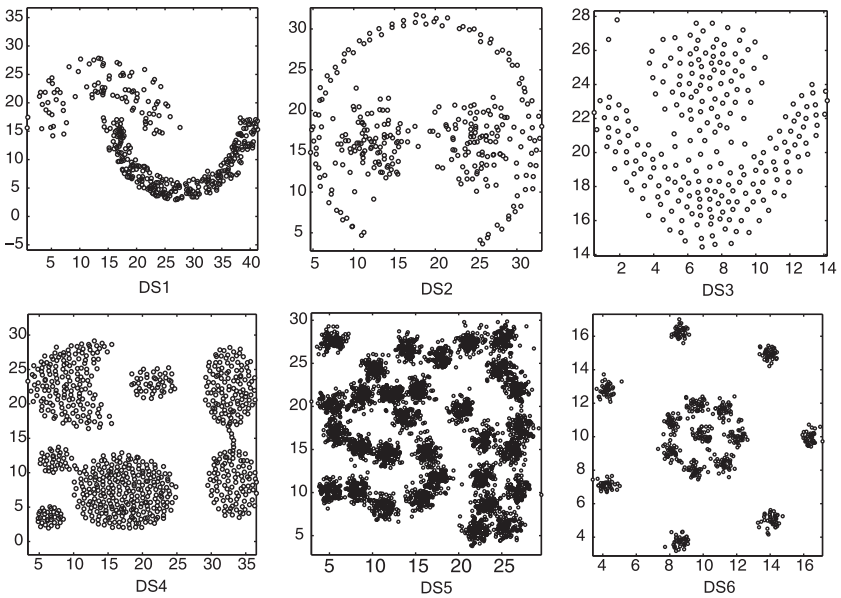

Fig. 5. Illustration of the six original synthetic datasets.

Table 1
Descriptions of used datasets.

Data set	Data size (N)	Dimensionality (d)	Number of clusters (K)
DS1	373	2	2
DS2	300	2	3
DS3	240	2	2
DS4	788	2	7
DS5	3100	2	31
DS6	600	2	15
Iris	150	4	3
Wine	178	13	3
WBC	683	9	2
WDBC	569	30	2

1. SAM-No-Pruning: split-and-merge without the pruning step.
2. SAM-2-MST: split-and-merge but using 2-MST instead of 3-MST.
3. SAM-SL: split-and-merge but using single linkage within the merge stage instead of the criterion based on $R()$-values.

As K-means and CSM may produce different partitions in different runs, in the following experiments we take the best clustering result out of 10 trial runs performed for each dataset in terms of one of the most used external clustering validity indices, Adjusted Rand (see Section 3.1). For DBScan, since we have no a priori knowledge about the parameters (*MinPts* and *Eps*), proper values were selected by trial and error.

3.1. External clustering validity indexes

An external clustering validity index is defined to measure the degree of correspondence between a clustering result and the prespecified partitions [41]. We will employ the four popular external indexes, Rand, FM, Jaccard, and Adjusted Rand ([34,41,44,48]), to evaluate the quality of the clustering results. Briefly, these four indexes are described as follows.

Suppose $\{P_1,\ldots,P_L\}$ is the set of prespecified partitions, $\{C_1,\ldots,C_K\}$ is a set of partitions produced by a clustering. For a pair of vectors $(\mathbf{x}_u,\mathbf{x}_v)$, it is referred as: (a) SS pair if $\exists I(\mathbf{x}_u \in P_I \wedge \mathbf{x}_v \in P_I)$ and $\exists J(\mathbf{x}_u \in C_J \wedge \mathbf{x}_v \in C_J)$, (b) SD pair if $\exists I(\mathbf{x}_u \in P_I \wedge \mathbf{x}_v \in P_I)$ and $\nexists J(\mathbf{x}_u \in C_J \wedge \mathbf{x}_v \in C_J)$, (c) DS pair if $\nexists I(\mathbf{x}_u \in P_I \wedge \mathbf{x}_v \in P_I)$ and $\exists J(\mathbf{x}_u \in C_J \wedge \mathbf{x}_v \in C_J)$, (d) DD pair if $\nexists I(\mathbf{x}_u \in P_I \wedge \mathbf{x}_v \in P_I)$ and $\nexists J(\mathbf{x}_u \in C_J \wedge \mathbf{x}_v \in C_J)$. Let a, b, c, d denote the numbers of SS, SD, DS, and DD pairs, respectively, and M the number of total pairs, the four indexes are defined as:

1. Rand

$$R = (a+d)/M. \tag{13}$$

2. Jaccard

$$J = a/(a+b+c). \tag{14}$$

3. FM

$$FM = \sqrt{\frac{a}{a+b}\frac{a}{a+c}}. \tag{15}$$

4. Adjusted Rand

$$AR = \frac{2(Ma-(a+b)(a+c))}{M(2a+b+c)-2(a+b)(a+c)}. \tag{16}$$

3.2. Results on synthetic datasets

DS1: This instance contains two datasets shaped like a crescent with different densities. The clustering results are illustrated in Fig. 6. SAM and SAM-No-Pruning, DBScan, CSM and CHAMELEON can partition the dataset properly. Both SAM-2-MST and SAM-SL fail. Since K-means favors spherical clusters, it fails on DS1. Single linkage clustering produces unsatisfactory partitions because it measures the distance between two clusters as the minimum distance between the pairs of data points in the two clusters. In the spectral clustering algorithm, the similarity matrix is created by a Gaussian kernel function with Euclidean distances, but the clustering result of this algorithm is similar to that of K-means. Although DBScan produces the expected partitions, it was difficult to tune the two parameters to achieve the proper result.

DS2: The set is composed of two Gaussian distributed clusters and one unclosed ring cluster surrounding the first two. Fig. 7 illustrates the clustering results. SAM and SAM-2-MST provide satisfactory clustering results, whereas SAM-No-Pruning

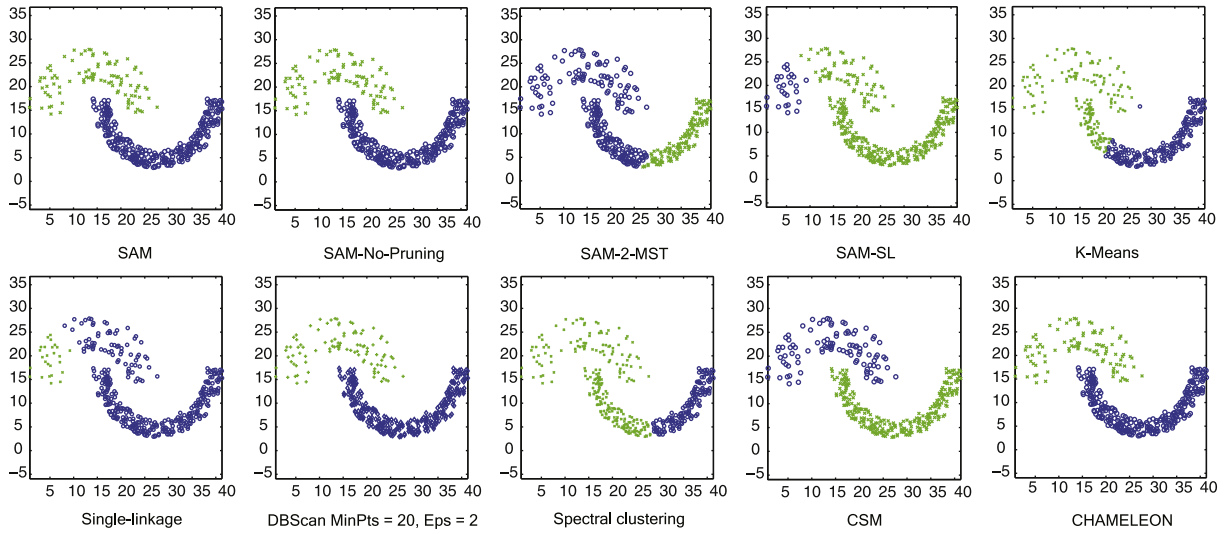

Fig. 6. Illustration of clustering results on DS1.

and SAM-SL produce improper results. CSM can sometimes identify the proper clusters, but not always. The results of DBScan and spectral clustering are not perfect, but better than those of *K*-means, single linkage, and CHAMELEON.

DS3: This dataset contains a spherical cluster and a half ring shaped cluster. The clustering results are shown in Fig. 8. All the algorithms except *K*-means and single linkage discover the expected clusters.

DS4: This dataset consists of 7 Gaussian distributed clusters. Fig. 9 illustrates the clustering results. SAM, its variant SAM-No-Pruning and SAM-2-MST, spectral clustering, CSM, and CHAMELEON find the expected clusters. SAM-SL and *K*-means provide partitions with low quality.

DS5: This dataset consists of 31 Gaussian distributed clusters. The clustering results are illustrated in Fig. 10. Spectral clustering, CSM, and CHAMELEON produce the expected partitions. SAM and SAM-SL are also successful except they fail to detect the cluster on the rightmost part. SAM-No-Pruning and SAM-2-MST perform much worse, whereas *K*-means, single linkage, and DBScan fail to detect almost all the expected clusters.

DS6: The 15 Gaussian distributed clusters in this dataset are arranged in two concentric circles. Fig. 11 describes the clustering results. SAM, SAM-SL, spectral clustering, CSM, and CHAMELEON produce the proper partitions, but SAM-No-Pruning, SAM-2-MST, *K*-means, single linkage, and DBScan do not.

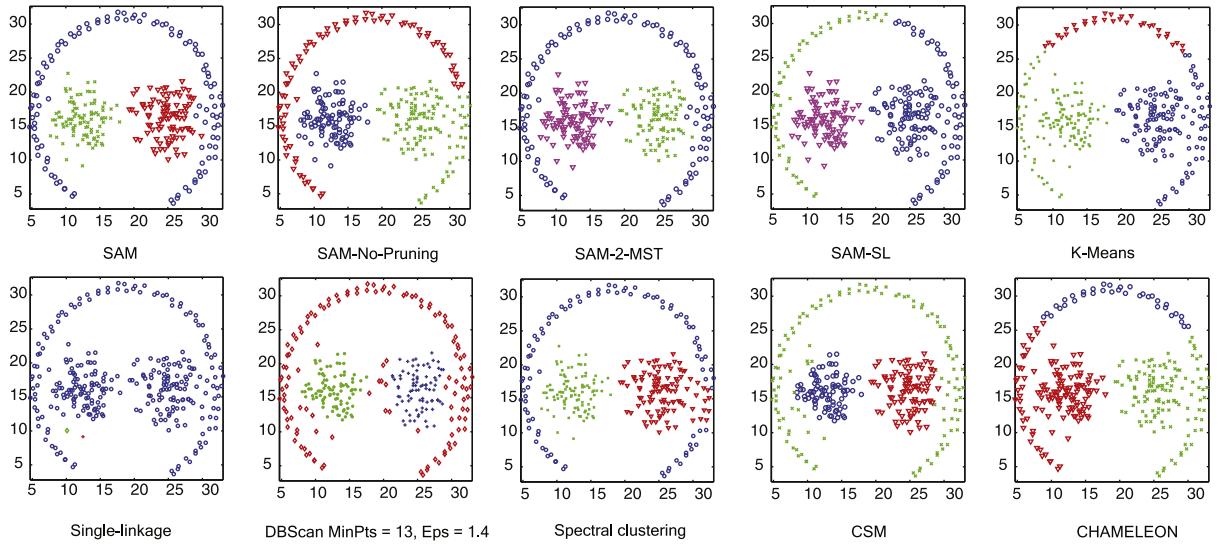

Fig. 7. Illustration of clustering results on DS2.

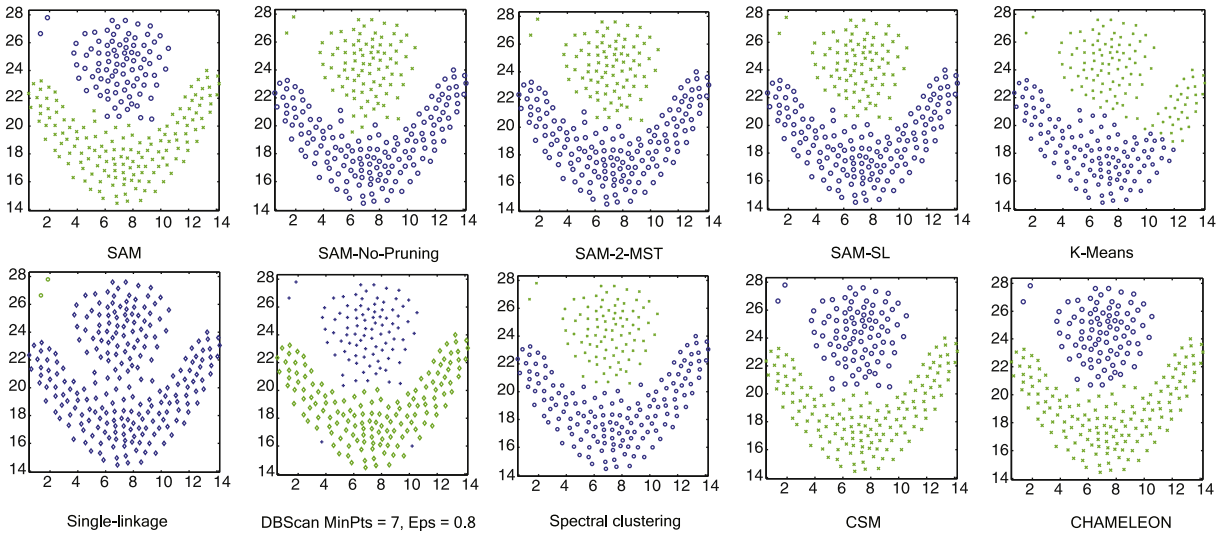

Fig. 8. Illustration of clustering results on DS3.

The corresponding Adjusted Rand index values of the clustering results on these six synthetic datasets are shown in Table 2.

3.3. Results on real datasets

The performance on each of the four UCI datasets is evaluated using the four common external clustering validity indices: Rand, Adjusted-Rand, Jaccard coefficient, and Fowlkes and Mallows (FM). The evaluation results on the four datasets are shown in Tables 3–6, respectively. The parameters of DBScan for IRIS, WINE, WBC, and WDBC are set to (MinPts = 8, Eps = 0.4), (MinPts = 3, Eps = 0.3), (MinPts = 9, Eps = 2), (MinPts = 4, Eps = 32.3), respectively.

For the IRIS dataset, Table 3 indicates that SAM, SAM-2-MST, and SAM-SL have the same performance and outperform the others. SAM-No-Pruning, CSM, DBScan, and *K*-means also provide partitions with relative high quality. In the case of the WINE dataset, the corresponding clustering qualities are shown in Table 4. *K*-means has the best performance, single linkage and spectral clustering provide more proper partitions than the proposed method SAM. It can be seen from Table 5 that SAM outperforms the other algorithms except *K*-means on the WBC dataset. As for the WDBC dataset in Table 6, DBScan has the

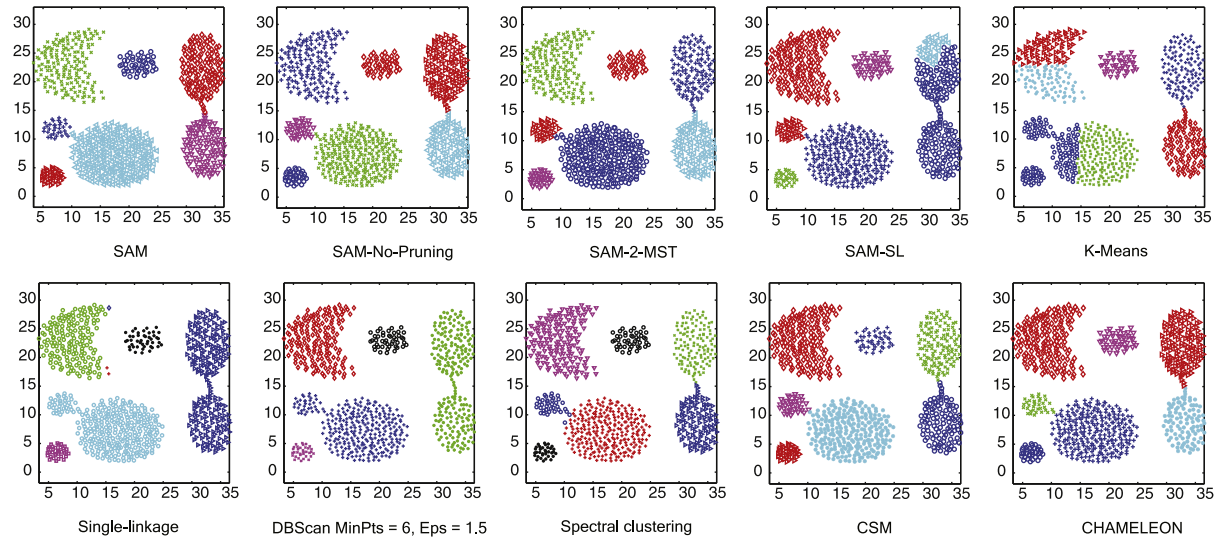

Fig. 9. Illustration of clustering results on DS4.

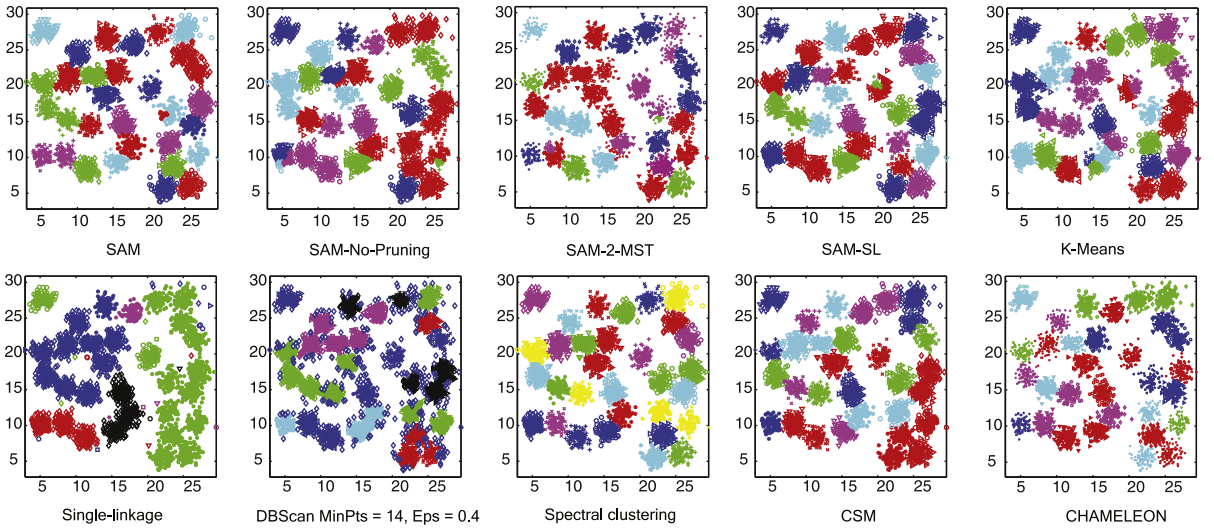

Fig. 10. Illustration of clustering results on DS5.

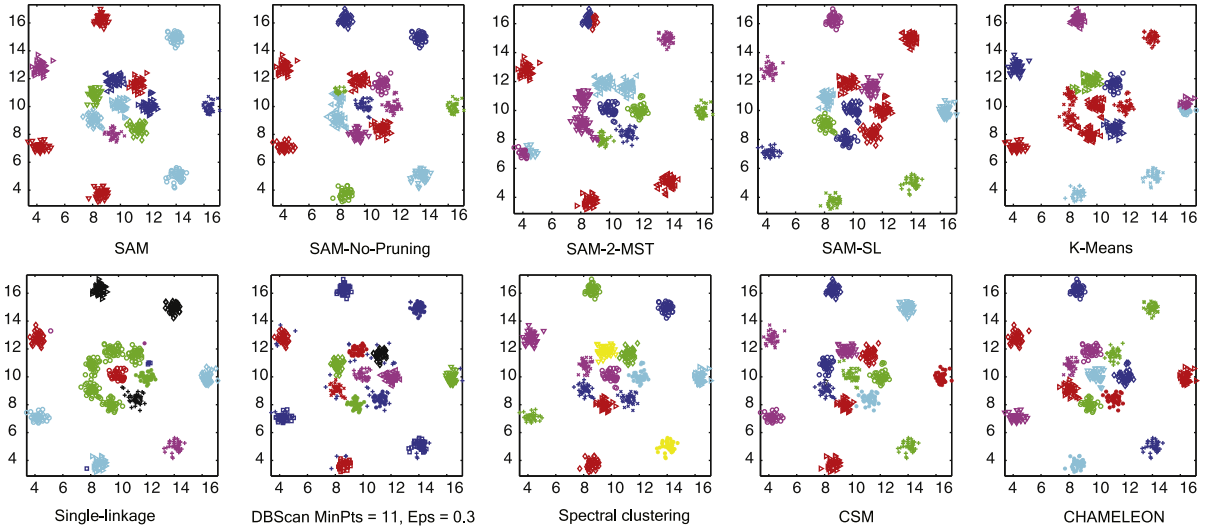

Fig. 11. Illustration of clustering results on DS6.

Table 2
Adjusted Rand index values of clustering performances on the six synthetic datasets.

Method	DS1	DS2	DS3	DS4	DS5	DS6
SAM	1.0000	0.9597	0.9339	0.9835	0.8896	0.9928
SAM-No-Pruning	1.0000	0.7272	0.9180	0.9920	0.8051	0.8723
SAM-2-MST	0.3181	0.9597	0.9178	0.9902	0.8183	0.8375
SAM-SL	0.2563	0.6130	0.9178	0.8743	0.8755	0.9928
K-means	0.5146	0.4739	0.4312	0.7186	0.8218	0.8055
Single linkage	0.2563	0.0004	0.0103	0.7996	0.1739	0.5425
DBScan	1.0000	0.8213	0.8859	0.8043	0.4321	0.8804
Spectral clustering	0.3178	0.8757	0.9178	0.9919	0.9522	0.9928
CSM	1.0000	0.9118	0.9667	0.9978	0.9196	0.9857
CHAMELEON	1.0000	0.4756	0.9617	1.0000	0.9274	0.9928

Table 3
Clustering performances on IRIS.

Method	Rand	Adjusted Rand	Jaccard	FM
SAM	0.9495	0.8858	0.8578	0.9234
SAM-No-Pruning	0.9075	0.7436	0.7298	0.8548
SAM-2-MST	0.9495	0.8858	0.8578	0.9234
SAM-SL	0.9495	0.8858	0.8578	0.9234
K-means	0.8797	0.7302	0.6959	0.8208
Single linkage	0.7766	0.5638	0.5891	0.7635
DBScan	0.8834	0.7388	0.7044	0.8268
Spectral clustering	0.8115	0.5745	0.5571	0.7156
CSM	0.8859	0.7455	0.7119	0.8321
CHAMELEON	0.7783	0.5492	0.5680	0.7369

Table 4
Clustering performances on WINE.

Method	Rand	Adjusted Rand	Jaccard	FM
SAM	0.7334	0.4007	0.4294	0.6009
SAM-No-Pruning	0.7012	0.3842	0.4113	0.5819
SAM-2-MST	0.7334	0.4007	0.4294	0.6009
SAM-SL	0.6976	0.3956	0.4702	0.6521
K-means	0.8797	0.7302	0.6959	0.8208
Single linkage	0.7766	0.5638	0.5891	0.7635
DBScan	0.6878	0.3171	0.3866	0.5582
Spectral clustering	0.7655	0.4741	0.4821	0.6506
CSM	0.6742	0.3757	0.4708	0.6618
CHAMELEON	0.7364	0.4769	0.5266	0.7049

Table 5
Clustering performances on WBC.

Method	Rand	Adjusted Rand	Jaccard	FM
SAM	0.9026	0.8033	0.8372	0.9114
SAM-No-Pruning	0.8876	0.7682	0.8061	0.8953
SAM-2-MST	0.8922	0.7820	0.8222	0.9025
SAM-SL	0.5565	0.0337	0.5453	0.7346
K-means	0.9240	0.8465	0.8703	0.9307
Single linkage	0.5453	0.0025	0.5444	0.7375
DBScan	0.8767	0.7529	0.7913	0.8838
Spectral clustering	0.5218	0.0246	0.4142	0.5867
CSM	0.5658	0.0585	0.5468	0.7333
CHAMELEON	0.5235	0.0279	0.5107	0.7007

best performance, while the proposed method SAM is better than *K*-means, single linkage, spectral clustering, CSM as well as its variants.

3.4. Discussion about the SAM variants

Pruning gives a small but consistent improvement (SAM vs. SAM-No-Pruning) based on the numeric results (Tables 2–5). In the case of DS2, pruning is critical to obtain the correct clustering whereas for the other test sets its effect is more like fine-tuning.

For the merge criterion, the index $R(C_i, C_j)$ is more complicated to evaluate than some simple alternatives like the single linkage criterion. Its effect on the clustering quality, however, is significant and in most cases the proposed approach (SAM) outperforms the single linkage variant (SAM-SL) using the single linkage algorithm in the merge stage. We consider the criterion-based merge critical for the performance of the algorithm.

The role of the *k*-MST is important. The value $k = 3$ was fixed already in preliminary tests using DS1-DS6, but let us discuss other choices. In SAM, 1-MST cannot be used in the merge stage as the criterion requires more information about the neighborhood than a simple spanning tree can provide, as the results of SAM-SL already showed. The question about the exact value of *k*, however, is less critical. In most cases, SAM and SAM-2-MST provide the same result, and only in two cases (DS1 and DS2) the 2-MST variant fails.

Table 6
Clustering performances on WDBC.

Method	Rand	Adjusted Rand	Jaccard	FM
SAM	0.8138	0.6269	0.6981	0.8223
SAM-No-Pruning	0.8003	0.6091	0.6772	0.8101
SAM-2-MST	0.8012	0.6190	0.6811	0.8197
SAM-SL	0.5308	0.0032	0.5219	0.7162
K-means	0.7504	0.4914	0.6499	0.7915
Single linkage	0.5326	0.0024	0.5315	0.7286
DBScan	0.8691	0.7367	0.7828	0.8782
Spectral clustering	0.7479	0.4945	0.6133	0.7604
CSM	0.5860	0.1335	0.5392	0.7201
CHAMELEON	0.8365	0.6703	0.7406	0.8514

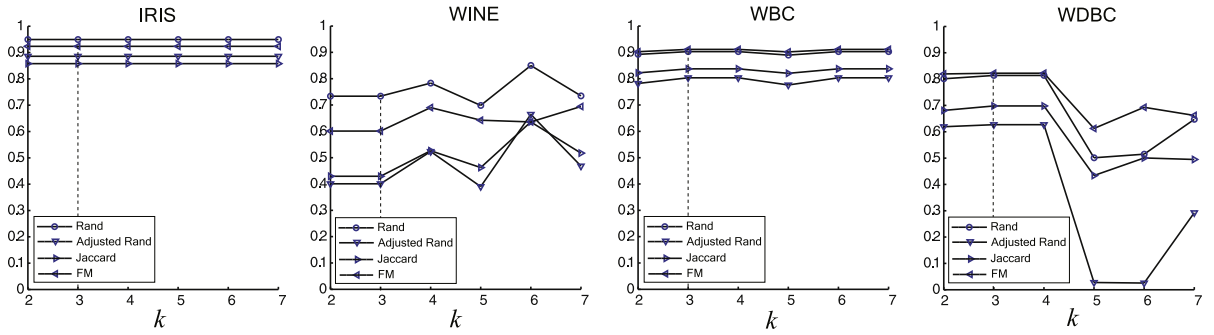

Fig. 12. The quality of clustering results for different values of k to compute k-MSTs.

Higher values than $k = 3$ were also tested, see Fig. 12. In most cases, the improvements due to higher values for k are insignificant and just increase the processing time. A contradicting example is the WDBC test set where higher values turn out to be harmful. Therefore, it is reasonable that the value of k is set to 3.

4. Conclusion

The proposed method employs minimum spanning trees in different stages. Before a dataset is split into different clusters, the hairs (leaves together with the connecting edges) of the first MST computed for the whole instance are pruned.

In the splitting stage, more than the desired K clusters are created by K-means. Three MSTs on an iteratively refined graph are computed and combined to determine the initial prototypes for K-means, because randomly selected initial prototypes would lead to unstable partitions.

After splitting, the initial MST is employed to adjust the partitions to make each subgroup corresponding to a subtree. Finally, in the merge step only neighboring pairs with respect to the same MST are considered to be merged.

Experiments have demonstrated the importance of each step of the algorithm. Except the number of clusters, there are no parameters left to be tuned by the user.

In summary, various MSTs are utilized during the whole split-and-merge algorithm because they can capture the intrinsic structure of a dataset. However, the computational complexity of constructing an MST is close to $O(N^2)$. The expensive computational cost obstructs the application of an MST to large scale data sets. One of our future work is to find a fast algorithm to construct an approximate MST.

One drawback of the proposed method is that the universality of the definitions of inter-connectivity and intra-similarity is insufficient. Although there does not exist a universal clustering method that can deal with all kinds of clustering problems, we try to improve the definition of inter-connectivity and intra-similarity to make the proposed method suitable for as many clustering problems as possible.

Although the proposed method does not have any direct restrictions for being applied to datasets with high dimensions, it is assumed to have all the same weaknesses as the other distance-based methods. Because to determine the intrinsic structure of this kind of datasets, different dimensions may have varied importance, whereas the proposed method equally considers all of dimensions of an input dataset. Thus, subspace clusterings [7,8] or other methods tailored for high dimensional data are expected to work better for high-dimensional datasets.

Acknowledgments

The authors thank the anonymous reviewers for their constructive comments and suggestions which helped improve the quality of this paper. The work of C. Zhong was supported by Zhejiang Provincial Natural Science Foundation of China, No.

Y1090851, and the Center for International Mobility (CIMO). The work of D. Miao was supported by the National Natural Science Foundation of China, No. 60970061, No. 61075056, and the Research Fund for the Doctoral Program of Higher Education, No. 20060247039.

References

[1] R. Agrawal, J. Gehrke, D. Gunopulos, P. Raghavan, Automatic subspace clustering of high dimensional data for datamining applications, in: Proceedings of ACM-SIGMOD Conference on the Management of Data, 1998, pp. 94–105.
[2] M. Bereta, T. Burczynski, Immune *k*-means and negative selection algorithms for data analysis, Inform. Sci. 179 (2009) 1407–1425.
[3] J.C. Bezdek, N.R. Pal, Some new indexes of cluster validity, IEEE Trans. Syst. Man Cybern. B. Cybern. 28 (1998) 301–315.
[4] H. Chang, D.Y. Yeung, Robust path-based spectral clustering, Pattern Recognit. 41 (2008) 191–203.
[5] D. Cheng, R. Kannan, S. Vempala, G. Wang, A divide-and-merge methodology for clustering, ACM Trans. Database Syst. 31 (2006) 1499–1525.
[6] M.C. Chiang, C.W. Tsai, C.S. Yang, A time-efficient pattern reduction algorithm for *k*-means clustering, Inform. Sci. 181 (2011) 716–731.
[7] Y. Chu, J. Huang, K. Chuang, D. Yang, M. Chen, Density conscious subspace clustering for high-dimensional data, IEEE Trans. Knowl. Data Eng. 22 (2010) 16–30.
[8] Y. Chu, Y. Chen, D. Yang, M. Chen, Reducing redundancy in subspace clustering, IEEE Trans. Knowl. Data Eng. 21 (2009) 1432–1446.
[9] T.H. Corman, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, second ed., MIT Press, Cambridge, MA, 2001.
[10] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial data sets with noise, in: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, Portland, OR, 1996, pp. 226–231.
[11] M. Figueiredo, A.K. Jain, Unsupervised learning of finite mixture models, IEEE Trans. Pattern Anal. Mach. Intell. 24 (2002) 381–396.
[12] P. Fränti, O. Virmajoki, V. Hautamäki, Fast agglomerative clustering using a k-nearest neighbor graph, IEEE Trans. Pattern Anal. Mach. Intell. 28 (2006) 1875–1881.
[13] L. Fu, E. Medico, FLAME, a novel fuzzy clustering method for the analysis of DNA microarray data, BMC Bioinformatics 8 (3) (2007).
[14] A. Gionis, H. Annila, P. Tsaparas, Clustering aggregation, ACM Trans. Knowl. Disc. Data 1 (2007) 1–30.
[15] S. Guha, R. Rastogi, K. Shim, CURE: an efficient clustering algorithm for large databases, in: Proceedings of the 1998 ACM-SIGMOD International Conference Management of Data (SIGMOD'98), 1998, pp. 73–84.
[16] S. Guha, R. Rastogi, K. Shim, ROCK: a robust clustering algorithm for categorical attributes, in: Proceedings of the IEEE Conference on Data Engineering, 1999, pp. 512–521.
[17] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, Data Mining, Inference and Prediction, Springer, New York, 2001.
[18] A. Hinneburg, D.A. Keim, An efficient approach to clustering in large multimedia databases with noise, in: Knowledge Discovery and Data Mining, 1998, pp. 58–65.
[19] C.-C. Hsu, C.-L. Chen, Y.-W. Su, Hierarchical clustering of mixed data based on distance hierarchy, Inform. Sci. 177 (2007) 4474–4492.
[20] A.K. Jain, M. Murthy, P. Flynn, Data clustering: a review, ACM Comput. Surv. 31 (1999) 264–323.
[21] A.K. Jain, R.C. Dubes, Algorithms for Clustering Data, Prentice-Hall, Englewood Cliffs, NJ, 1988.
[22] A.K. Jain, M.C. Law, Data clustering: a user's dilemma, in: Pattern Recognition and Machine Intelligence, LNCS, vol. 3776, Springer-Verlag, Berlin, Heidelberg, 2005, pp. 1–10.
[23] G. Karypis, E.H. Han, V. Kumar, CHAMELEON: a hierarchical clustering algorithm using dynamic modeling, IEEE Trans. Comput. 32 (1999) 68–75.
[24] L. Kaufman, P.J. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis, Wiley, New York, 1990.
[25] T. Kaukoranta, P. Fränti, O. Nevalainen, Iterative split-and-merge algorithm for VQ codebook generation, Opt. Eng. 37 (1998) 2726–2732.
[26] B. King, Step-wise clustering procedures, J. Am. Stat. Assoc. 69 (1967) 86–101.
[27] J.Z.C. Lai, T.J. Huang, An agglomerative clustering algorithm using a dynamic *k*-nearest-neighbor list, Inform. Sci. 181 (2011) 1722–1734.
[28] J.S. Lee, S. Olafsson, Data clustering by minimizing disconnectivity, Inform. Sci. 181 (2011) 732–746.
[29] C.H. Lee, O.R. Zaïane, H.-H. Park, J. Huang, R. Greiner, Clustering high dimensional data: a graph-based relaxed optimization approach, Inform. Sci. 178 (2008) 4501–4511.
[30] C.R. Lin, M.S. Chen, Combining partitional and hierarchical algorithms for robust and efficient data clustering with cohesion self-merging, IEEE Trans. Knowl. Data Eng. 17 (2005) 145–159.
[31] M. Liu, X. Jiang, A.C. Kot, A multi-prototype clustering algorithm, Pattern Recognit. 42 (2009) 689–698.
[32] J. MacQueen, Some methods for classification and analysis of multivariate observations, in: Fifth Berkeley Symposium on Mathematics, Statistics and Probability, 1967, pp. 281–297.
[33] G. McLachlan, K. Basford, Mixture Models: Inference and Application to Clustering, Marcel Dekker, New York, 1988.
[34] G.W. Milligan, S.C. Soon, L.M. Sokol, The effect of cluster size, dimensionality, and the number of clusters on recovery of true cluster structure, IEEE Trans. Pattern Anal. Mach. Intell. 5 (1983) 40–47.
[35] R.T. Ng, J. Han, CLARANS: a method for clustering objects for spatial data mining, IEEE Trans. Knowl. Data Eng. 14 (2002) 1003–1016.
[36] M.R. Rezaee, B.P.F. Lelieveldt, J.H.C. Reiber, A new cluster validity index for the fuzzy *c*-mean, Pattern Recognit. Lett. 19 (1998) 237–246.
[37] S.E. Schaeffer, Graph clustering, Comput. Sci. Rev. 1 (2007) 27–64.
[38] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 22 (2000) 888–905.
[39] P.H.A. Sneath, R.R. Sokal, Numerical Taxonomy, Freeman, San Francisco, London, 1973.
[40] G.T. Toussaint, The relative neighborhood graph of a finite planar set, Pattern Recognit. 12 (1980) 261–268.
[41] S. Theodorodis, K. Kouttoumbas, Pattern Recognition, fourth ed., Academic Press, Amsterdam, 2009.
[42] C.J. Veenman, M.J.T. Reinders, E. Backer, A maximum variance cluster algorithm, IEEE Trans. Pattern Anal. Mach. Intell. 24 (2002) 1273–1280.
[43] W. Wang, J. Yang, M. Muntz, STING: a statistical information grid approach to spatial datamining, in: Proceedings of the International Conference on Very Large Data Bases, 1997, pp. 186–195.
[44] M.J. Warrens, On the equivalence of Cohen's kappa and the Hubert–Arabie Adjusted Rand index, J. Classif. 25 (2008) 177–183.
[45] Z. Wu, R. Leahy, An optimal graph theoretic approach to data clustering: theory and its application to image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 15 (1993) 1101–1113.
[46] R. Xu, D. Wunsch II, Survey of clustering algorithms, IEEE Trans. Neural Netw. 16 (2005) 645–678.
[47] Y. Xu, V. Olman, D. Xu, Clustering gene expression data using a graph-theoretic approach: an application of minimum spanning tree, Bioinformatics 18 (2002) 536–545.
[48] K.Y. Yeung, C. Fraley, A. Murua, A.E. Raftery, Model-based clustering and data transformations for gene expression data, Bioinformatics 17 (2001) 977–987.
[49] C.T. Zahn, Graph-theoretical methods for detecting and describing gestalt clusters, IEEE Trans. Comput. C-20 (1971) 68–86.
[50] <http://www.ics.uci/mlearn/MLRespository.html>.