A New Clustering Algorithm Based on Graph Connectivity

Yu-Feng Li[1], Liang-Hung Lu[1], and Ying-Chao Hung[2(✉)]

[1] Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan
lhlu@cc.ee.ntu.edu.tw

[2] Department of Statistics, National Chengchi University, Taipei 11605, Taiwan
hungy@nccu.edu.tw

Abstract. A new clustering algorithm based on the concept of graph connectivity is introduced. The idea is to develop a meaningful graph representation for data, where each resulting sub-graph corresponds to a cluster with highly similar objects connected by edge. The proposed algorithm has a fairly strong theoretical basis that supports its originality and computational efficiency. Further, some useful guidelines are provided so that the algorithm can be tuned to optimize the well-designed quality indices. Numerical evidences show that the proposed algorithm can provide a very good clustering accuracy for a number of benchmark data and has a relatively low computational complexity compared to some sophisticated clustering methods.

Keywords: Clustering · Graph theory · Time complexity

1 Introduction

Finding meaningful clusters in large and complex data has been an important task with a wide range of applications in biology, medicine, economics, psychology, etc. This type of analysis belongs to the so-called unsupervised learning, and the goal is to partition data into a certain (usually small) number of groups with homogeneous/similar objects [1]. Over the years, there are a bunch of clustering techniques introduced in the literature, which includes the hierarchical clustering (e.g. agglomerative [2,3] and divisive trees [4]), squared error-based optimization (e.g. K-means [5,6] and K-medoids [7]), mixture densities-based approaches [8], graph theory [9,10], fuzzy analysis [11], neural networks (e.g. self-organizing maps [12]), kernel-based approaches [13], just to name a few.

The usefulness of high connectivity in graphs to cluster analysis dates back to the works in [14,15]. A similar concept was developed to represent the hierarchical clustering [16,17]. Note that graph theory can also be used for nonhierarchical clustering. Specifically, the idea is to discard inconsistent edges in the minimum spanning tree and treat the remaining connected objects as clusters [7].

K. Arai et al. (Eds.): SAI 2018, AISC 858, pp. 442–454, 2019.
https://doi.org/10.1007/978-3-030-01174-1_33

A remarkable work is to treat the clusters as Highly Connected Subgraphs (HCS) and employ a minimum cut procedure to separate a graph [18]. A probability version for constructing the HCS is implemented by the CLICK algorithm, which places Gaussian assumptions on the clusters and assigns probability weights to edges [19]. Another similar work is the CAST algorithm, which models clusters as corrupted clique graphs and adds high affinity objects or removes low affinity objects from the cluster until no more changes occur [20]. Recently, a number of graph clustering algorithms are developed to integrate structural and attribute similarities - the readers can refer to [21,22] for some remarkable works.

In this paper, we propose a novel clustering algorithm based on the concept of graph connectivity. Our goal is to identify a set of "noise objects" (named as *singular points* in this work) so that meaningful subgraphs can be created to recover all clusters in data. In addition, the algorithm is designed to have relatively low computational complexity in comparison with some sophisticated clustering algorithms introduced in the literature.

The remaining of this work is organized as follows. In Sect. 2, we introduce some fundamental notations in graph theory and describe how the algorithm is developed step by step. In addition, we provide (i) theoretical results to justify the rationale of using identified singular points for finding meaningful clusters and (ii) guidelines for choosing all tuning parameters in the algorithm and the best number of clusters. In Sect. 3, we briefly introduce the time and storage complexity of our proposed algorithm. In Sect. 4, we illustrate our algorithm on some selected benchmark data and evaluate its performance by comparing with some well-known clustering algorithms in terms of clustering accuracy, time complexity, and storage complexity. Conclusions are made in Sect. 5.

2 Clustering Algorithm Based on Graph Connectivity

Consider a set of n objects $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ in $\mathbb{R}^m$, and the objective is to find a set of meaningful clusters $C = \{C_1, C_2, \ldots\}$, where each object $x_i \in C_j$ for some j, $C_i \cap C_j = \emptyset$ for $i \neq j$, and $\mathcal{X} = \cup_j C_j$. Objects belonging to the same cluster are considered as *homogeneous* or *similar*. Note that to exclude trivial cases, here we assume (i) the number of objects n is fairly large and (ii) the number of clusters $|C|$ is relatively smaller than n. We next introduce some notations in graph theory that are necessary for developing our new clustering algorithm.

2.1 Notations in Graph Theory

In graph theory each object x_j is treated as a *vertex* (or node), while an *edge* is used to connect similar objects. The idea of clustering is to partition the data into a small number of edge-disjoint *sub-graphs*, where each sub-graph refers to one cluster with similar objects connected by edges. An illustration of clustering based on graph connectivity is given in Fig. 1, where data are partitioned into three edge-disjoint sub-graphs G_1, G_2, and G_3. Note that if we use a graph

notation G to represent the data $\mathcal{X}$ in Fig. 1, then we can write $G = G_1 \cup G_2 \cup G_3$. Further, the vertices in G_1, G_2, G_3 represent the objects in three disjoint clusters C_1, C_2, and C_3, respectively.

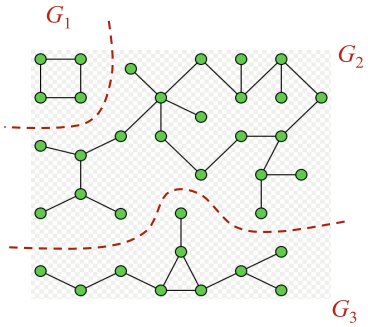

Fig. 1. An illustration of clustering based on graph connectivity with three edge-disjoint sub-graphs G_1, G_2 and G_3.

To develop a meaningful graph (as shown in Fig. 1), for each object $x_i \in \mathcal{X}$ our first step is find the closest k objects (in Euclidean distance) and make connections with these k objects (by k edges). This will automatically form an *undirected* graph $G^k = (V, E)$, where V represents the set of all vertices (i.e. all objects) and E represents the set of all connected edges.

2.2 Selecting the Number of Edges for Graph Connection

Note that graph G^k is obviously affected by the choice of k. For example, if k is chosen to be a smaller number, then every object is less likely to connect to other objects. This will tend to create a larger number of sub-graphs (or clusters). On the other hand, if k is chosen to be a larger number, then every object is more likely to connect to other objects. This will tend to create a smaller number of clusters. However, since our algorithm can further split the graph in later stages, in the first step we recommend choosing a larger value of k. The following is a rule of thumb for choosing k, which is based on the topological property (e.g., properties of a hexagon in a 2-D space) and a large number of experimental trials:

$$k \geq 6(m-1), \quad \text{where } m \text{ is the data dimension.} \tag{1}$$

For example, in a two dimensional data space ($m = 2$), we will recommend choosing the value of k to be at least 6.

2.3 Identifying Singular Points for Splitting the Graph

It is noted that controlling the value of k may not always succeed in finding a meaningful clustering, especially when the clusters have overlapping objects.

These overlapping objects will often create "bad edges" to connect the graph, thus resulting in a rather small number of clusters. We next introduce a procedure for identifying these potential noise objects (named as *singular points* in this paper), which are the key for splitting the graph.

Once we obtain the graph $G^k = (V, E)$ based on a selected value of k, for any object $x_i = v \in V$ we define the *set of distance-d vertices* by

$$V_v(d) = \{u \in V : d_{G^k}(u, v) = d\}, \tag{2}$$

where $d_{G^k}(u, v)$ is the *geodesic distance* between u and v based on graph G^k, viz., the minimum number of edges connecting u and v in G^k. A geometric illustration of the set $V_v(d)$ is given in Fig. 2, where v is an object located in the middle of a dumbbell-shape data and d is chosen to be 8 and 10.

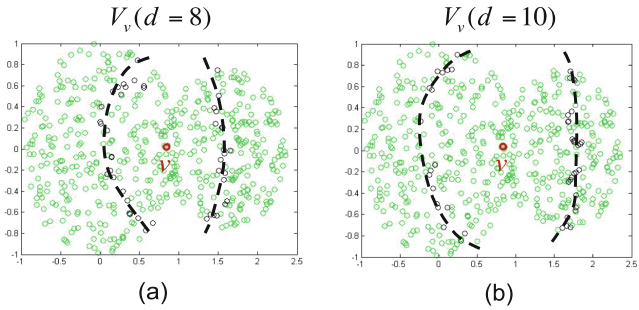

Fig. 2. A geometric illustration of the set $V_v(d)$ (points on the black dashed lines) for an object v located in the middle of a dumbbell-shape data, and d is chosen to be 8(a) and 10(b).

Then, we make connections between two vertices $u_1, u_2 \in V_v(d)$ if $d_{G^k}(u_1, u_2) \leq d$. This will create a new graph $G^k_{v,d} = (V_v(d), E_v(d))$, for which the resulting set of clusters is denoted by $C|G^k_{v,d}$. We proceed by introducing some other notations. Let us denote the set of singular points we wish to identify by S and the graph with all singular points being excluded by $G_{-S} = (V \setminus S, E)$. For a positive integer a, define also the graph $G^k_{v,d,a} = (V_v(d), E_v(d, a))$, where the set of edges $E_v(d, a) = \{\{u_1, u_2\} : d_{G^k}(u_1, u_2) \leq a \text{ and } u_1, u_2 \in V_v(d)\}$. Similarly, the resulting set of clusters is denoted by $C|G^k_{v,d,a}$. Now, we are ready for showing the following theorem:

Theorem 1: Given k, v, d, $G^k_{v,d}$, $C|G^k_{v,d}$ and assume: (i) $|V_v(d)| \gg |C|G^k_{v,d}|$ and (ii) all clusters $C_i \in C|G^k_{v,d,a}$ have similar sizes, then there exists some positive integer $a \leq 2d$ such that

$$SI(v|k, d, a) = \sum_{i:C_i \in C|G^k_{v,d,a}} \frac{1 + \log|C_i \cap V_v(d)|}{\log\left(|V_v(d)| + 1\right)}$$
$$\approx |C|G^k_{v,d,a}|. \tag{3}$$

Proof: We briefly sketch the proof here. Given any $v \in V$, it is clear that

$$\sum_{i:C_i \in C|G^k_{v,d,a}} [1 + \log|C_i \cap V_v(d)|] \approx \sum_{i:C_i \in C|G^k_{v,d,a}} \log [r_i \cdot |V_v(d)|], \tag{4}$$

where $0 \leq r_i \leq 1$ for all i and $\sum_{i:C_i \in C|G^k_{v,d,a}} r_i = 1$. Since we assume $|V_v(d)| \gg |C|G^k_{v,d}|$ and $r_i \cong r = |C|G^k_{v,d,a}|^{-1}$ for all i, (4) can be approximately written as

$$\begin{aligned} & \frac{1}{r} \cdot \{\log r + \log |V_v(d)|\} \\ = \; & |C|G^k_{v,d,a}| \cdot \left\{\log|C|G^k_{v,d,a}|^{-1} + \log |V_v(d)|\right\} \\ = \; & |C|G^k_{v,d,a}| \cdot \log|C|G^k_{v,d,a}|^{-1} + |C|G^k_{v,d,a}| \cdot \log |V_v(d)| \\ \approx \; & |C|G^k_{v,d,a}| \cdot \log (|V_v(d)| + 1). \end{aligned} \tag{5}$$

Dividing (4) by the term $\log (|V_v(d)| + 1)$ in (5), we then obtain

$$SI(v|k, d, a) \approx |C|G^k_{v,d,a}|.$$

For an appropriately defined set of singular indices S, we can show that $|C|G^k_{v,d,a}| \approx |C|G^k_{-S}|$, which is the number of clusters by excluding all the singular points from data. This implies that the value of $SI(v|k, d, a)$, named as the *singular index*, can be used to identify if v is a potential "noise object". More specifically, if $SI(v|k, d, a) > 1$, we say that object v is a singular point since there exists "more than one cluster" a geodesic distance d away from v (which is the result indicated by Theorem 1). Note that once all singular points are identified, we can develop the graph G^k_{-S} (by excluding all singular points) and find out the associated clustering result $C|G^k_{-S}$. To cluster each singular point $s \in S$, we can simply employ the "plurality vote" (or weighted vote) based on the clustering of vertices in the *neighborhood* of s. Here the neighborhood is defined via the geodesic distance based on graph G^k.

Figure 3 illustrates all the identified singular points for the dumbbell data shown in Fig. 2, where $(k, a) = (6, 5)$ and $d = 8$ and 10.

It should be mentioned that different choices of d and a may result in different sets of singular points (recall that k is fixed beforehand). As can be seen from Fig. 3, a smaller value of d results in fewer singular points and a larger value of d (but not unbounded) results in more singular points. This will nevertheless affect the clustering result. Here we provide some useful guidelines for choosing the values of d and a. Based on a large number of data clustering experiences, our first guideline suggests that

$$5 \leq d \leq diam(G^k), \tag{6}$$

where $diam(G^k)$ is the maximum geodesic distance between two vertices in graph G^k. The second guideline suggests that one can find the best combination of

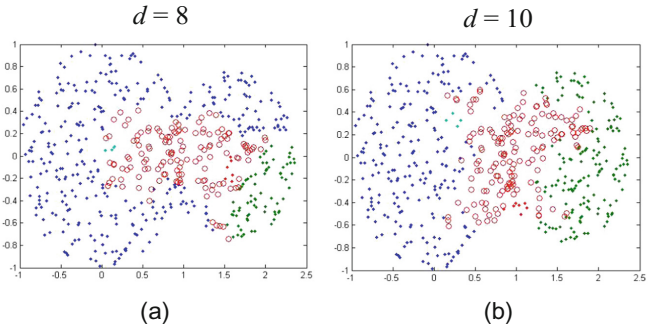

Fig. 3. An illustration of identified singular points (in red color) for the dumbbell data shown in Fig. 2, where $(k, a) = (6, 5)$ and $d = 8$(a) and 10(b).

(d, a) so that the variation of singular indices for all identified singular points is minimized. That is,

$$(d^*, a^*) = \underset{\substack{5 \le d \le diam(G^k) \\ 0 < a < 2d}}{\arg} \min \mathrm{Var}(SI(v|k, d, a)). \tag{7}$$

Note that (7) is clearly a mixed-integer optimization problem in a constrained 2-dimensional space. Therefore, finding the optimum solution (d^*, a^*) is not a difficult task, it can be done by a simple grid search.

2.4 Determining the Number of Clusters

Since each singular index, under some conditions, is approximately the number of clusters around a singular point, it can be used to determine the best number of clusters for the entire data set. Once (k, d, a) are appropriately chosen based on the guidelines introduced above, we can collect the singular indices of all identified singular points and examine the likelihood (or probability) that each possible number of clusters happens. For example, Fig. 4 shows the histograms of all singular indices greater than one (after taking the ceiling function) for the dumbbell data shown in Fig. 2, where $(k, d, a) = (6, 8, 5)$ and $(6, 10, 5)$. As can be seen, both histograms suggest there are two clusters in the data.

Another way of choosing the best number of clusters is to maximize the ratio of "between-group variance S_B" to "within-group variance S_W" by controlling the values of (d, a), that is,

$$\frac{\mathrm{Trace}(S_B)}{|C| - 1} \Big/ \frac{\mathrm{Trace}(S_W)}{n - |C|}. \tag{8}$$

Based on the empirical study given by [23], this strategy achieves the best performance among 30 criteria for determining the number of clusters. We summarize the detailed steps of our clustering method in the following algorithm.

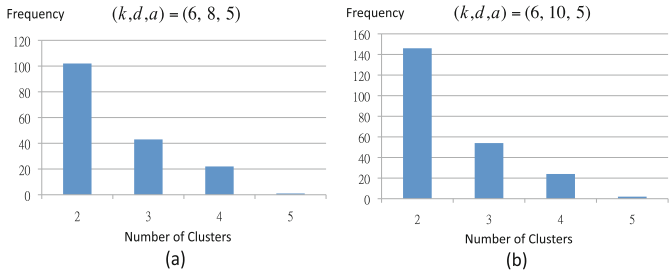

Fig. 4. The histograms of all singular indices greater than one (by taking the ceiling function) for the dumbbell data shown in Fig. 2, where $(k, d, a) = (6, 8, 5)$ (panel (a)) and $(k, d, a) = (6, 10, 5)$ (panel (b)).

2.5 Algorithm

(1) Initial Setup: Compute all pairwise Euclidean distances $||x_i - x_j||$, $x_i, x_j \in \mathcal{X}$, $i, j \in \{1, \ldots, n\}$.

Step 1: Select a fairly large k based on the rule in (1), develop the graph $G^k = (V, E)$ based on all data in $\mathcal{X}$.

Step 2: Select a value d based on (6) and a value $a < 2d$, compute the singular index $SI(v|k, d, a)$ by (3) for each $v \in G^k$.

Step 3: Repeat Step 2 and search the best combination (d^*, a^*) by utilizing the guideline given in (7).

Step 4: Identify the set of singular points S based on the selected (k, d^*, a^*), create a graph $G^k_{-S} = (V \setminus S, E)$ by excluding all singular points.

Step 5: Re-cluster $V \setminus S$ based on graph G^k_{-S} and assign clusters to each singular point $s \in S$ by plurality vote (or weighted vote), obtain the clustering result C.

Note that one may determine the best number of clusters in C by examining the histograms of $SI(v|k, d^*, a^*)$ (as shown in Fig. 4) and the guideline given in (8). This allows to adequately revise the choice of (d^*, a^*) in Step 3 and execute Steps 4–5 one more time so as to obtain a better clustering result.

3 Time and Space Complexity

Let us describe first the space complexity of the proposed algorithm. First, the space complexity for the initial setup is $O(n^2)$, since we need to store all pairwise distances of the n objects. Then, in Step 1 the space complexity for storing the n vertices of the graph $G^k = (V, E)$ is clearly $O(n)$. These together then constitute a total space complexity $O(n^2)$.

Next, we explore the time complexity for implementing the proposed algorithm. First, the time complexity for obtaining the nearest k neighbors of each object is $O(n \log n)$, which is based on the structure of a k-d tree [24]. Next, the time complexity for establishing the graph G^k is $O(n)$, which is obtained by

using the Breadth-First Search [25] and triangle inequality for geodesic distance. We now examine the time complexity for identifying all the singular points. Let D be the matrix that stores all pairwise geodesic distances between vertices in G^k and denote its inverse by $D^{-1} = I$ (we can show that I always exists). The reason why we employ the matrix I for implementing the algorithm is the following: If $d_{G^k}(u_1, u_2) = \delta$, we then have

$$I(u_1^*, \delta) = u_2 \quad \text{and} \quad I(u_2^*, \delta) = u_1, \tag{9}$$

where u_1^* and u_2^* are the two "pointers" referring to the original indices of objects u_1 and u_2 in G^k, respectively. Note that (9) can be used to speed up the search of $V_v(d)$, which results in the time complexity $O\left(\sum_{v\in V} V_v(d)\right)$. Further, by placing an assumption that clusters in $V_v(d)$ are fairly uniformly distributed, we can show that

$$\begin{aligned} O\left(\sum_{v\in V} V_v(d)\right) &\approx O\left(n\left(\frac{d}{d_{max}}\sqrt[m]{n}\right)^{m-1}\right) \\ = O\left(\left(\frac{d}{d_{max}}\right)^{m-1} n^{2-1/m}\right) &= O\left(r_d^{m-1} n^{2-1/m}\right), \end{aligned} \tag{10}$$

where $r_d = d/d_{max} < 1$ and $d_{max} = diam(G^k)$. Since $m \geq 2$ and n is large, the time complexity of implementing the algorithm is obviously dominated by (10).

4 Clustering Benchmark Data: Illustration and Evaluation

In this section, we first demonstrate how to utilize the proposed algorithm for clustering some benchmark data. Then, we evaluate the algorithm by comparing with other well-known and widely used clustering algorithms in terms of accuracy, time complexity, and space complexity.

4.1 Banana Sets

We illustrate our clustering algorithm on two selected data with banana shapes. In the first data the two bananas are located a bit far from each other, while in the second data the two bananas are intentionally placed fairly close to each other. As discussed fruitfully in the literature, clustering this type of data appears to be a non-trivial task. The clustering results along with the identified singular points are shown in Figs. 5 and 6.

As can be seen from Figs. 5 and 6, our algorithm works very well for the first banana set. Obviously, there are two clusters found and we need merely a small number of singular points to separate two clusters. On the other hand, the two bananas have a certain number of overlapping points in the second set.

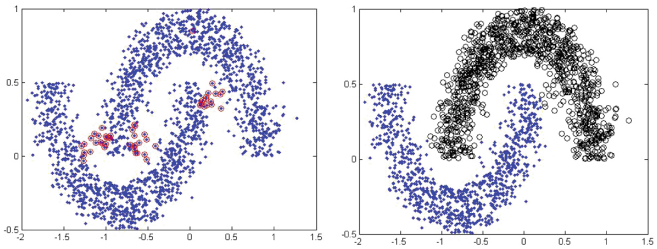

Fig. 5. The identified singular points (in red color) for the banana data (a) and the clustering result (b). Here we choose $(k, d, a) = (6, 5, 7)$.

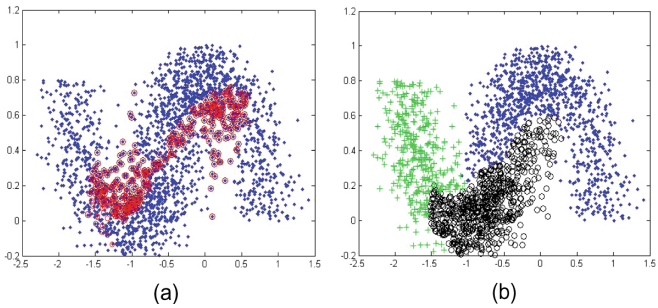

Fig. 6. The identified singular points (in red color) for the banana data (a) and the clustering result (b). Here we choose $(k, d, a) = (6, 15, 20)$.

In this case, we need to select the tuning parameters (k, d, a) so as to create enough singular points for separating different clusters. However, overloading the singular points may result in a larger number of clusters, as can be seen from Fig. 6 (there are 3 clusters found).

4.2 Ring Sets

We illustrate our clustering algorithm on a data set with two different sizes of rings. The clustering result is shown in Fig. 7. As can be seen, our algorithm works perfectly for clustering this ring data. In general, methods based on the graph theory works particularly well for this type of data with circular and curved structures.

4.3 Gaussian Clusters

We illustrate our clustering algorithm on a data emulated by 15 bivariate Gaussian distributions. The result is shown in Fig. 8. As can be seen, our algorithm also works perfectly for this type of data with a fairly large number of clusters.

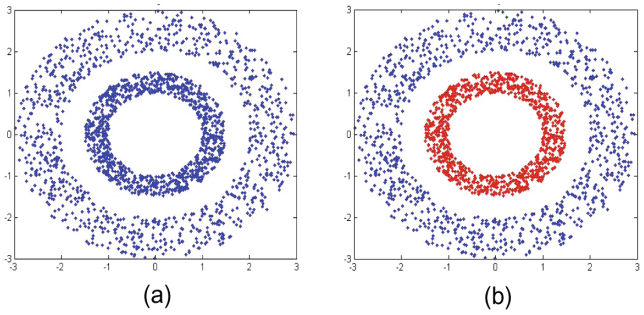

Fig. 7. The original data comprised of two clusters with a ring structure (a) and the clustering result (b). Here we choose $(k, d, a) = (6, 5, 5)$.

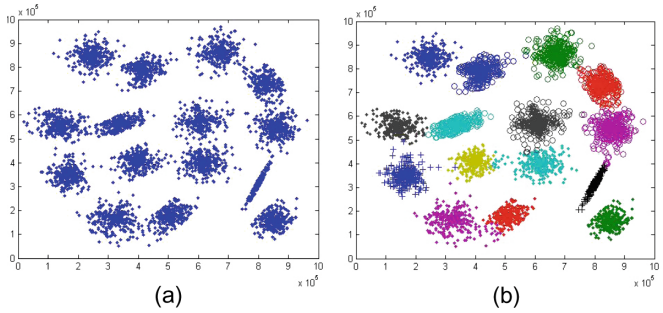

Fig. 8. The original data comprised of 15 clusters emulated by 15 bivariate Gaussian distributions (a) and the clustering result (b). Here we choose $(k, d, a) = (6, 9, 7)$.

We next evaluate the numerical performance of our proposed algorithm by comparing with other well-known clustering algorithms - such as the K-means, kernel K-means, and another graph-based method called the Highly Connected Sub-graphs (HCS) implemented by the well-known efficient Stoer-Wagner algorithm [26]. The clustering accuracy along with the time and storage complexity are provided for all methods based on three benchmark data: Iris, banana set, and ring set. It is noted that the Iris data have four features and three labeled classes. To implement the clustering algorithms, we hide the class labels and treat each class as a cluster. The results are given in Table 1.

As can be seen from Table 1, our algorithm has a very good clustering accuracy for all three benchmark data considered in this study - the accuracy is high above 90%. The two methods kernel K-means and HSC also have a clustering accuracy above 90% for the banana and ring data, but do not perform as well as our algorithm for the Iris data (the accuracy is below 90% for both methods). It is worth noting that our algorithm has the time complexity $O(r_d^{m-1} n^{2-1/m})$. Since $m > 1$ and $r_d < 1$, it is obviously less than that of kernel K-means and HSC (they are $O(n^2)$ and $O(n^2 \log n)$, respectively). To see how much time complexity is reduced by our algorithm in practice, let us consider an example with $m = 5$,

Table 1. The clustering accuracy, time complexity, and storage complexity for the methods considered in this study. Note that the "Big O" notation is removed due to a limited space

Data	Our Algorithm	K-means	Ker. K-means	HSC
Iris	91%	78%	89%	85%
Banana	91%	55%	93%	92%
Ring	98%	54%	93%	95%
Time	$r_d^{m-1} n^{2-1/m}$	n	n^2	$n^2 \log n$
Space	n^2	n	n^2	n^2

$r_d = 0.5$, and $n = 1,000$. Some algebra shows that the kernel K-means and HCS require respectively 10^6 and 6.9×10^6 computations, while our algorithm requires merely 1.57×10^4 computations - a staggering two orders of magnitude reduction. It is also noted that the K-means algorithm can be developed (e.g. the Lloyd's Algorithm [5]) to have a low time and storage complexity $O(n)$. However, it has a relatively low clustering accuracy (78%, 55% and 54%) for these three benchmark data. In summary, compared to the well-known clustering methods, our algorithm appears to be very competitive in clustering accuracy and has a relatively low computational complexity (or time complexity).

5 Conclusion

In this paper, we propose a new clustering algorithm based on the concept of graph connectivity. We also provide some useful guidelines so that all parameters in the algorithm can be adequately tuned so as to produce a meaningful clustering result. Numerical evidences show that our algorithm can provide a very good clustering accuracy for a number of benchmark data. In addition, it has a relatively low time complexity in comparison with two sophisticated clustering methods - kernel K-means and HCS. It should be mentioned that due to the conditions placed in Theorem 1, the idea of employing the singular points for splitting the data may not succeed in some cases - e.g., the data with unbalanced sizes of clusters or with a small number of objects compared to the feature dimension (i.e. data with a small n and relatively large m). We are currently investigating (i) how to improve the algorithm so that it can deal with data with various types of clustering structures; and (ii) the performance of our algorithm in comparison with other sophisticated clustering algorithms. We hope to report some promising results in the future work.

References

1. Hansen, P., Jaumard, B.: Cluster analysis and mathematical programming. Math. Program. **79**, 191–215 (1997)
2. Sneath, P.: The application of computers to taxonomy. J. Gen. Microbiol. **17**, 201–226 (1957)
3. Sorensen, T.: A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyzes of the vegetation on danish commons. Biologiske Skrifter **5**, 1–34 (1948)
4. Kaufman, L., Rousseeuw, P.: Finding Groups in Data: An Introduction to Cluster Analysis. Wiley, New York (1990)
5. Lloyd, S.: Least squares quantization in PCM. IEEE Trans. Inf. Theory **28**, 129–137 (1982)
6. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer, New York (2001)
7. Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice-Hall, New Jersey (1988)
8. Everitt, B., Landau, S.: Cluster Analysis. Amold, London (2001)
9. Harary, F.: Graph Theory. Addison-Wesley, Boston (1969)
10. Karypis, G., Han, E., Kumar, V.: Chameleon: hierarchical clustering using dynamic modeling. IEEE Comput. **32**, 68–75 (1999)
11. Zadeh, L.: Fuzzy sets. Inf. Control **8**, 338–353 (1965)
12. Kohonen, T.: Self-organizing Maps. Springer, New York (2001)
13. Schölkopf, B., Smola, A., Müller, K.: Nonlinear component analysis as a kernel eigenvalue problem. Neural Comput. **10**, 1299–1319 (1998)
14. Matula, D.W.: The cohesive strength of graphs. In: Chartrand, G., Kapoor, S.F. (eds.) The Many Facets of Graph Theory. Lecture Notes in Mathematics, vol. 110, pp. 215–221. Springer, Berlin (1969)
15. Matula, D.W.: k-components, clusters and slicings in graphs. SIAM J. Appl. Math. **22**, 459–480 (1972)
16. Cherng, J., Lo, M.: A hypergraph based clustering algorithm for spatial data sets. In: Proceedings of the IEEE International Conference on Data Mining, pp. 83–90 (2001)
17. Estivill-Castro, V., Lee, I.: AMOEBA: hierarchical clustering based on spatial proximity using Delaunay diagram. In: Proceedings of the 9th Symposium on Spatial Data Handling, pp. 7a.26–7a.41 (1999)
18. Hartuv, E., Shamir, R.: A clustering algorithm based on graph connectivity. Inf. Process. Lett. **76**, 175–181 (2000)
19. Sharan, R., Shamir, R.: CLICK: a clustering algorithm with applications to gene expression analysis. In: Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology, pp. 307–316 (2000)
20. Ben-Dor, A., Shamir, R., Yakhini, Z.: Clustering gene expression patterns. J. Comput. Biol. **6**, 281–297 (1999)
21. Zhou, Y., Cheng, H., Yu, J.X.: Graph clustering based on structural/attribute similarities. Proc. VLDB Endow. **2**, 718–729 (2009)
22. Parimala, M., Lopez, D.: Graph clustering based on structural attribute neighborhood similarity (SANS). In: Proceedings of the IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT), pp. 1–4 (2015)
23. Milligan, G., Cooper, M.: An examination of procedures for determining the number of clusters in a data set. Psychometrika **50**, 159–179 (1985)

24. Blum, M., Floyd, R.W., Pratt, V.R., Rivest, R.L., Tarjan, R.E.: The bounds for selection. J. Comput. Syst. Sci. **7**, 448–461 (1973)
25. Moore, E.F.: The shortest path through a maze. In: Proceedings of the International Symposium on the Theory of Switching, pp. 285–292 (1959)
26. Stoer, M., Wagner, F.: A simple min-cut algorithm. J. ACM **44**, 585–591 (1997)