Autonomous clustering by fast find of mass and distance peaks

Jie Yang, and Chin-Teng Lin, *Fellow, IEEE*

Abstract—Clustering is a fundamental tool of scientific analysis, ubiquitous in disciplines from biology and chemistry to astronomy and pattern recognition. We propose a novel clustering algorithm based on the natural idea that a cluster and its nearest neighbor with higher mass should be merged into one cluster, unless they both have relatively large masses and the distance between them is also relatively large. The find of mass and distance peaks reveals the mergers that don't conform to the rule and should be removed. The algorithm is parameter-free and harnesses this idea to recognize any cluster and find the proper number of clusters and noise autonomously. Experiments on numerous synthetic and real-world data sets show the enormous versatility of the proposed algorithm that remarkably outperforms the best compared algorithm. Additionally, we also compare it with latest state-of-the-art deep clustering algorithms on several challenging image data sets. The proposed algorithm without any deep representation achieves better or close performance than deep clustering algorithms on image clustering.

Index Terms—Clustering, parameter-free, data analysis, unsupervised learning

1 Introduction And Related work

GROUPING similar objects to derive insights from classes of things is a fundamental tool in the search for knowledge. It is used in virtually all the natural and social sciences and plays a central role in biology, astronomy, psychology, medicine, and chemistry [1]. Like many disciplines, grouping objects in data science is called clustering, and, as one of the three broadest categories of machine learning algorithms, clustering in one form or another is the important method of learning from unlabeled data.

Yet, despite the importance and ubiquity of clustering, and the plethora of existing algorithms, the current clustering methods suffer from a variety of drawbacks [1]. Much work has been done to overcome, circumvent, or minimize these problems. Many strategies are targeted, many are ingenious, and several tackles more than one problem. Yet none dispense with enough issues to be considered a universal clustering choice, because optimal clustering is typically an NP-hard problem. Often this means, researchers and analysts must test and tune several alternatives to determine which best suits their needs.

To run through the list of issues and remedies [2]–[4], standard hierarchical clustering has a high computational cost and a typical time complexity of $O(n^3)$; hence, these algorithms are not suitable for large-scale data sets. Further, stopping the clustering process requires some manually-determined condition, such as "Stop at k number of clusters". Some of the newer algorithms circumvent these shortcomings by either using a fast approximate nearest neighbor method to accelerate the clustering process [5]–[8] or by pruning the cluster trees to estimate the correct number of clusters [9], [10].

Partition clustering, such as K-means [11], demands that the number of clusters must either be known or estimated in advance. Also, these algorithms cannot detect non-convex clusters of varying size or density, and most of them are highly sensitive to noise, outliers, and getting the initialization phase "right". Improvements over these algorithms include the X-means algorithm [12], which is able to estimate the number of clusters automatically, and kernel K-means and variants, which solve the non-convex problem [13], [14]. Still further methods address the initialization problem [15], [16].

In the past, density clustering typically required a suite of thresholds to be set in advance – for example, the cutoff distance used to calculate the density of points, the number of points at which a cluster is deemed to be high density, and so on. Today's density-based algorithms employ a variety of tricks and techniques so as to depend less on these thresholds [17]–[25]. For example, Liang et al. [19] proposed the 3DC clustering that can automatically determine the number of clusters. Du et al. [25] introduced k nearest neighbors (KNN) and principal component analysis (PCA) into density peak clustering to estimate the number of clusters more accurately and achieve better results on high-dimensional data sets.

Model-based clustering generally relies on prior knowledge of many parameter settings, such as the distribution of each cluster, even though this information is often very difficult to acquire in practice. Hence, solutions to alleviate this disadvantage have also emerged [26]–[28]. For example, Scrucca, L. et al. [26] proposed an improved model-based clustering based on data transformations, which can lead to improved model fitting and more accurate clustering results. O'Hagan, A. et al. [27] proposed the Bayesian initialization averaging method to generate high-quality initial parameter settings for the expectation–maximization algorithm.

- *Jie Yang is with Computational Intelligence and Brain Computer Interface Lab, Australian AI Institute, FEIT, University of Technology Sydney, Australia (e-mail: jie.yang.uts@gmail.com).*
- *Chin-Teng Lin is with Computational Intelligence and Brain Computer Interface Lab, Australian AI Institute, FEIT, University of Technology Sydney, Australia (e-mail: Chin-Teng.Lin@uts.edu.au).*

Lastly, classic grid clustering also depends on many user-provided parameters, such as interval values to divide space and density thresholds, and the most algorithms do not scale to high-dimensional data sets. Many variant algorithms were proposed to solve these problems [29]–[31]. For example, Chen, J. et al. [29] proposed a new grid based clustering algorithm for hybrid data stream, which can automatically determine the number, center, and radius of clusters.

From this list, a good solution would therefore: be able to recognize all kinds of clusters regardless of shape, size or density; be parameter-free; not depend on a priori knowledge; have a relatively low computational overhead and a reasonable time complexity; be robust to noise and outliers; not require initialization; be able to automatically determine the number of clusters; and preclude the need for a manually-specified stopping condition.

To achieve these goals, we propose a novel clustering algorithm, called Torque Clustering (TC), based on the idea that a cluster and its nearest neighbor with higher mass should be merged into one cluster, unless they both have relatively large masses and the distance between them is also relatively large.

The merger process of this algorithm is inspired by the gravitational interactions when galaxies merge. In previous studies, the evolution of galaxies was described as a hierarchical process by astronomers using galaxy merger trees [32]–[35]. Galaxy mergers can occur when two or more galaxies come close enough to each together, and can be classified into two types due to their comparative size of the merging galaxies, including minor mergers and major mergers. According to the predictions of merger rates of dark matter haloes, minor mergers are expected to be much more common than major mergers [36]. TC simulates the process of galaxy minor mergers, so that clusters with larger masses continuously merge adjacent clusters with smaller masses. Similarly, the TC algorithm generates a hierarchical tree that reflects the natural structure of the data set.

After generating the hierarchical tree, TC estimates the correct number of clusters by pruning the cluster tree. However, unlike the existing methods based on probability density [9], [10], TC reveals the reasonable partition by the find of mass and distance peaks. In Newton's law of universal gravitation $\frac{Gm_1m_2}{r^2}$, set G as a constant aside for the moment and consider m_1m_2 and r^2. Regard m_1 and m_2 as the number of samples in two data clusters, and r^2 as the distance between them. TC exploits the two properties, m_1m_2 and r^2, to describe the merger of each pair of clusters. As a result, reasonable partitions can be obtained by removing the mergers with relatively larger m_1m_2 and r^2.

We evaluated TC on 20 data sets across five different domains: image recognition, biology, medicine, physics, and astronomy, and 19 state-of-the-art algorithms were included in the experiment for performance comparison. In terms of accuracy, TC ranked No. 1 on 15 of the 19 data sets (except for one without ground-truth labels), outperforming the best compared algorithm by a factor of 4 in average ranking. Additionally, TC returned the exact number of ground-truth clusters on 15 of the 20 data sets compared to the best prior automatic clustering algorithm with perfect accuracy on only 10. Moreover, we conducted an additional comprehensive evaluation on 56 data sets with noise, outliers, overlaps, imbalance, and high dimensionality. TC still retained a great performance advantage on these data sets. Finally, we also compared TC with latest state-of-the-art deep clustering algorithms on several challenging image data sets. Interestingly, TC without any deep representation can achieve better or close performance than deep clustering algorithms on image clustering.

2 THE ALGORITHM

Loosely based on conventional hierarchical clustering structures [37], the TC algorithm generates a hierarchical tree that reflects the natural structure of the data set. However, unlike most existing hierarchy-based algorithms, TC reaches higher accuracy with a significantly smaller number of merger steps and is robust to noise and outliers. In addition, no manual stopping condition is required; the final number of clusters does not need to be defined in advance; the density of each data sample does not need to be estimated, nor does the distribution of each cluster; and the feature space does not need to be divided into distributions. The comparison in mass and distance governs the merger process of TC, while the find of mass and distance peaks reveals the mergers that should be removed to leave a reasonable cluster partition. The TC algorithm is described in sections 2.1-2.7 below.

2.1 Define Clusters and Form Connections between Them

Consider a data set denoted as $X = \{x_1, x_2, \dots, x_n\}$, where $x_i \in R^D$. The first step is to determine the initial "mass" of the data set, which is simply the number of samples. Thus, initially, each data sample x_i is considered to be its own cluster ζ_i, which yields an initial cluster set of $\Gamma = \{\zeta_1, \zeta_2, \dots, \zeta_n\}$. This forms the first layer of the hierarchical tree. A count of the set gives us the initial mass, denoted as $\Theta = \{\theta_1, \theta_2, \dots, \theta_n\}$. At this initial step, each $\theta_i = 1$. The following rule is then applied to form connections between clusters:

$$\zeta_i \to \zeta_i^N, \quad if\ \theta_i \le \theta_i^N, \tag{1}$$

where ζ_i^N denotes the 1-nearest cluster to ζ_i, and θ_i^N denotes the number of samples ζ_i^N contains. The symbol " $\to$ " denotes the connection (i.e., merger) C_i between ζ_i and ζ_i^N. Regarding each cluster as a vertex, then a connected graph G can be obtained.

A new set of clusters Γ' can be formed by

$$\Gamma' = \Phi(G), \tag{2}$$

where $\Phi(\,)$ identifies the samples contained in each connected component as a new cluster.

Then, applying Eq. (1) to Γ' generates new connections C_i, which alters the connected graph G. Eq. (2) on G generates the next cluster set Γ', and the cycle continues until there is only one all-encompassing cluster at the top of a hierarchical tree. Different from the classic agglomerative

hierarchical clustering, this constrained method of merging avoids yielding undesirable elongated clusters and can be performed in parallel as long as two neighboring clusters satisfy the requirement of Eq. (1). Additionally, this exercise of treating each data sample as a cluster and taking the set through a series of mergers until all have fully merged does three things: 1) It builds a hierarchical map of partitioning clusters at different granularities; 2) it provides a map for the algorithm to choose the most appropriate clustering scheme; and 3) it gives analysts the choice to manually override the automatic selections and choose a different granularity if desired. This leaves the question of what criteria the algorithm uses to determine the "most appropriate" scheme. Based on the above idea, if a connection has both a relatively large mass and stretches over a long distance, it is "abnormal"; removing it should reveal a more reasonable partition structure. The following steps set out the mechanisms for detecting and removing abnormal connections.

2.2 Define Two Properties of Each Connection to Construct the Decision Graph

Abnormal connections can be identified by observing two intuitive properties of the connection C_i. One of the properties is the product of the mass value of the two clusters it connects

$$M_i = \theta_i \times \theta_i^N; \quad (3)$$

the other is the square of the distance between the two clusters it connects

$$D_i = d^2(\zeta_i, \zeta_i^N). \quad (4)$$

Plotting all the connections on a two-dimensional graph of the two properties, called a decision graph, will reveal that the mass and distance of the abnormal connections are abnormally larger than, and further away from, those of the normal connections. Fig. 1 provides an example to illustrate the core idea of the proposed TC algorithm.

There are many studies on defining the distance between two clusters [38]. Here, we simply measure the nearest distance from any member of one cluster to any member of the other cluster as the distance between the two clusters, i.e., $d(\zeta_i, \zeta_i^N) = \min_{x_a \in \zeta_i, x_b \in \zeta_i^N} d(x_a, x_b)$. With large-scale data, a fast approximate nearest neighbor method like k-d tree or locality-sensitive hashing may be a more appropriate choice to search nearest cluster since the distance computation approach used in these methods negates the need to actually know the distances between any two clusters [7], [8]. Further, the computation costs would be low and the complexity could be kept to $O(nlog(n))$ as compared to the complexity of standard hierarchical clustering algorithms, which is $O(n^3)$. In this way, the proposed TC algorithm is highly scalable. A detailed analysis of the time and space complexity of TC is presented in section 2.6.

The detail of how TC works is best explained through an example, which is set out step-by-step in Figure 2 and Table 1.

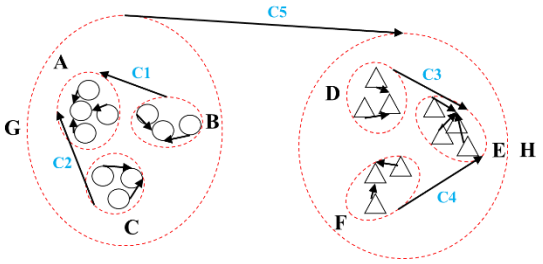
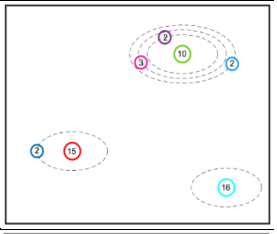


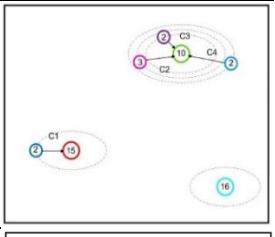
Figure 1. The core idea of the proposed algorithm. The red dotted lines delineate the clusters A-H in this two-dimensional data distribution, derived from Eq. (1) and Eq. (2). The black lines C1-C5 indicate the connections from each cluster to its nearest cluster with a length of L_i, where L_5 is the longest. Each cluster is at one end of a connection C_i, and contains several samples. For example, the clusters A and E each have four samples, the clusters B, C, D, and F each have three samples, and the clusters G and H each have 10 samples. Our goal is to find abnormal connections, which are defined as those with both a relatively large distance (i.e., D_i Eq. (4)) and a relatively large number of samples (i.e., M_i Eq. (3)). Obviously, connection C_5, with 10 samples in each of the clusters it connects, is carrying the greatest mass and it is the longest L_5. But what is key is the relativity. C_5 is unique in that it is markedly longer than the other sets of connections. Removing C_5 and calculating the connected components according to Eq. (2) results in a final, more reasonable, set of clusters. This approach is consistent with human intuition as well as the natural laws of gravitational interactions.

Figure 2. A step-by-step example of how TC works. Consider a data set where, initially, each sample has a mass of 1 and is treated as its own cluster. Applying Eq. (1) to each cluster establishes connections between clusters, resulting in a connected graph. Applying Eq. (2) to the graph, new clusters begin to emerge (indicated in different-colored circles) with a mass equal to the number of samples within them (the value in the circles).

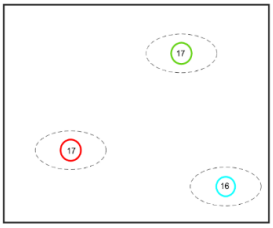
A
Applying Eq. (2) to the connected graph of the initial clusters reveals seven new larger clusters.



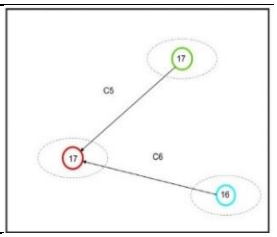
B
Connections C_1-C_4 can then be added according to the adjacency relationship given by Eq. (1). Now, the two properties M_i and D_i of C_1-C_4 can be calculated according to Eqs. (3)- (4), as shown in Table 1a.



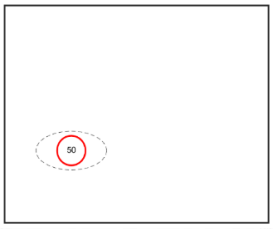
C
Again, applying Eq. (2) to the connected graph of Fig. 2B reveals three new larger clusters. The mass of each new cluster is equal to the sum of the masses of the sub-clusters it contains.



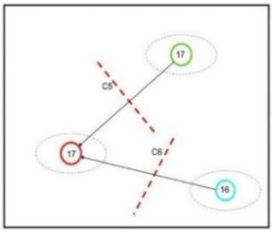
D
Connections C_5-C_6 can then be added according to the adjacency relationship given by Eq. (1). Now, the two properties M_i and D_i of C_5-C_6 can be calculated Eqs. (3)- (4), as shown in Table 1b.



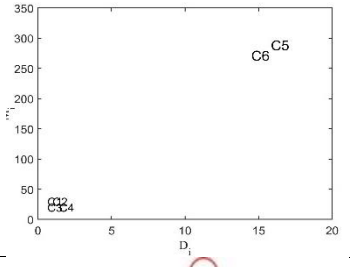
E
Again, applying Eq. (2) to the connected graph of Fig. 2D, we now have one big cluster, and the merging process is complete. Steps A-E show that this process has established a hierarchical tree of clustering partitions at different granularities, as illustrated in Fig. 2H.



F
Returning to Fig. 2D for a moment, it is easy to see from the decision graph in Fig. 2G that the relative maxima of the two properties that need to be removed are at C_5, C_6. These connections are identified as abnormal, as indicated by the red dotted lines.



G
Plotting all six connections on a two-dimensional graph of the properties, i.e., the decision graph, indeed shows that C_5, C_6 are abnormally further away and larger than $C_1 - C_4$.



H
Hence, connections C_5, C_6 are removed to arrive at the final partitioning scheme. This entire clustering process can be represented as a hierarchical tree, as the dendrogram to the right shows. The solid black arrows indicate the calculation of connected components, i.e., Eq. (2).

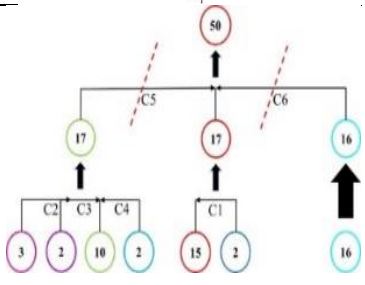


TABLE 1A. PROPERTIES OF THE CLUSTERS AND CONNECTIONS IN FIGS. 2A AND 2B.

Cluster	Mass of the cluster	Nearest cluster	Mass of the Nearest cluster	Connect it or not?	M_i	D_i	Connect number
	15		2	N	–	–	–
	2		15	C	30	0.64	C1
	10		2	N	–	–	–
	3		10	C	30	1.00	C2
	2		10	C	20	0.64	C3
	2		10	C	20	1.44	C4
	16		15	N	–	–	–

TABLE 1B. PROPERTIES OF THE CLUSTERS AND CONNECTIONS IN FIGS. 2C AND 2D.

Cluster	Mass of the cluster	Nearest cluster	Mass of the Nearest cluster	Connect it or not?	M_i	D_i	Connect number
	17		16	N	-	-	-
	17		17	C	289	15.83	C5
	16		17	C	272	14.50	C6

Different colors correspond to different clusters. "N" means that the cluster does not connect to its nearest cluster, and "C" means it does connect. M_i and D_i are the two properties of the connection C_i.

2.3 Define the Torque of Each Connection and Sort the Connections in Descending Order

The decision graph produced by TC provides an efficient visualization tool to determine and remove abnormal connections. However, because a manual inspection of the 2D plot would be both prone to error and time-consuming, we propose an automatic method to determine abnormal connections based on a metric to indicate the gaps between connections. We call this metric the Torque Gap (*TGap*) because of its similarity in mathematical expression. The *TGap* is calculated by first calculating the torque τ_i of all connections, where

$$\tau_i = M_i \times D_i. \quad (5)$$

Obviously, if the two clusters connected by a connection have relatively large masses and the distance between them is relatively large, the τ_i of the connection must also be large.

Then we sort all connections in descending order according to their corresponding torque values, and call it the torque sorted connections list (TSCL). The connection in TSCL and its torque are denoted as $\grave{C}_i$ and $\grave{\tau}_i$, respectively.

According to our core idea above, abnormal connections must be the top several connections in the TSCL, because they have the largest torque values among all the connections. But how to specify the "several" ones? This requires us to calculate *TGap* on TSCL.

2.4 Define Torque Gap and Find the Largest Gap to Determine Abnormal Connections

The $TGap_i$ between each connection along with its following connection in the TSCL is calculated next. The formula for computing $TGap_i$ is

$$TGap_i = \omega_i \frac{\grave{\tau}_i}{\grave{\tau}_{i+1}}, \grave{\tau}_{i+1} \neq 0 \quad (6)$$

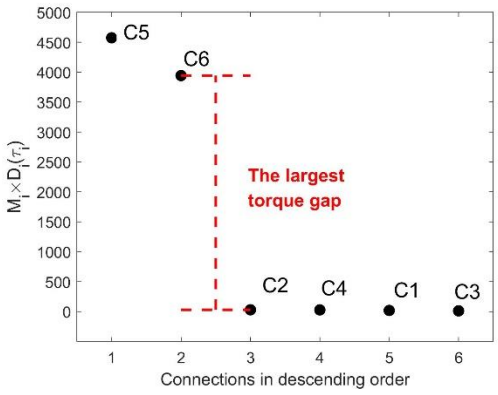


Fig. 2I. After calculating the torque of each connection generated from Figs.2A-E, all connections are sorted in the order of decreasing torque to obtain the torque sorted connection list (TSCL, i.e., $C_5, C_6, C_2, C_4, C_1, C_3$). Furthermore, Eqs. (6)-(9) are applied to find the largest torque gap between two adjacent connections in the TSCL (i.e., the torque gap between C_6 and C_2). As a result, C_5 and C_6 are regarded as the abnormal connections to be removed.

where ω_i is a weighted value that indicates the proportion of connections among the top i connections of TSCL that have relatively large M_i, D_i, and τ_i values.

The process for defining ω_i is as follows: Eq. (1) will reveal many connections C_i throughout the entire clustering process and, as we know, each C_i has two properties, M_i and D_i. Therefore, the set of connections that have relatively large M_i, D_i, and τ_i values among all the connections (denoted as $Large\_C$) can be defined as:

$$Large\_C = \{C_j | (\tau_j \geq mean\_\tau) \cap (M_j \geq mean\_M) \cap (D_j \geq mean\_D)\} \quad (7)$$

where $mean\_\tau$ is the mean value of all τ_i, $mean\_M$ is the mean value of all M_i, and $mean\_D$ is the mean value of all D_i.

Top_C_i is the set of the top i connections of TSCL, and can be defined as

$$Top\_C_i = \{\hat{C}_1, \hat{C}_2, \dots, \hat{C}_i\} \quad (8)$$

Based on $Large\_C$ and Top_C_i, ω_i is defined as:

$$\omega_i = \frac{|Large\_C \cap Top\_C_i|}{|Large\_C|} \quad (9)$$

The largest $TGap_i$ is denoted as $TGap_L$, and the L connections at the top of the TSCL (i.e., $\{\hat{C}_1, \hat{C}_2, \dots, \hat{C}_L\}$) are regarded as abnormal connections to be removed.

The $TGap$ in Eq. (6) considers two important factors in determining the abnormal connections at the same time: $\frac{\hat{\tau}_i}{\hat{\tau}_{i+1}}$ is the natural torque gap between adjacent connections in TSCL, and ω_i represents the natural clustering resolution. The larger the torque gap, the more suitable it is as a cutting point. The purpose of calculating $\frac{\hat{\tau}_i}{\hat{\tau}_{i+1}}$ is to find a fracture between the connections with larger torque values and the connections with smaller torque values among the sorted connections. ω_i exists to defend against the uneven distribution or imbalance of clusters in data set. For example, there is a data set containing three relatively balanced ground-truth clusters, {A}, {B}, and {C}. After performing TC on this data set, we get the connection between A and B, and the connection between B and C, denoted as C_{AB} and C_{BC}, respectively. Suppose the distance between A and B is much larger than the distance between B and C, and the distance between B and C is much larger than the distances between sub-clusters in A, B, and C. If we only rely on $\frac{\hat{\tau}_i}{\hat{\tau}_{i+1}}$ to determine the abnormal connections, it is likely to just remove C_{AB} to get the partition: {A}, {B, C}. However, we can consider both $\frac{\hat{\tau}_i}{\hat{\tau}_{i+1}}$ and ω_i at the same time. Since the M_i, D_i, and τ_i of C_{BC}, are also large relative to other connections except C_{AB} (i.e., $\tau_i \geq mean\_\tau, M_i \geq mean\_M, D_i \geq mean\_D$), then TC is more likely to remove both C_{AB} and C_{BC} to get the correct partition, {A}, {B}, and {C}.

2.5 Define Halo Connections to Determine the Noise

In cluster analysis, noise detection is also an important step. In our algorithm, we define another type of connection to determine the noise, which is called "halo connections" (denoted as Halo_C). Halo connections are characterized by a relatively large D_i and a relatively small M_i. The formula for computing halo connections is

$$Halo\_C = \left\{C_k | (M_k \leq mean\_M) \cap (D_k \geq mean\_D) \cap \left(\frac{D_k}{M_k} \geq mean\_\frac{D}{M}\right)\right\} \quad (10)$$

where the $mean\_\frac{D}{M}$ is the mean value of all $\frac{D_i}{M_i}$.

In section 2.4, after removing the L abnormal connections, L+1 clusters can be obtained. In this step, the halo connections are further removed, and then some small sub-clusters in the L+1 clusters can be found, which are considered as part of the cluster halo (suitable to be considered as noise). Fig. 3 provides an example to illustrate the power of abnormal connections and halo connections. The pseudocode of TC is provided in Algorithm 1.

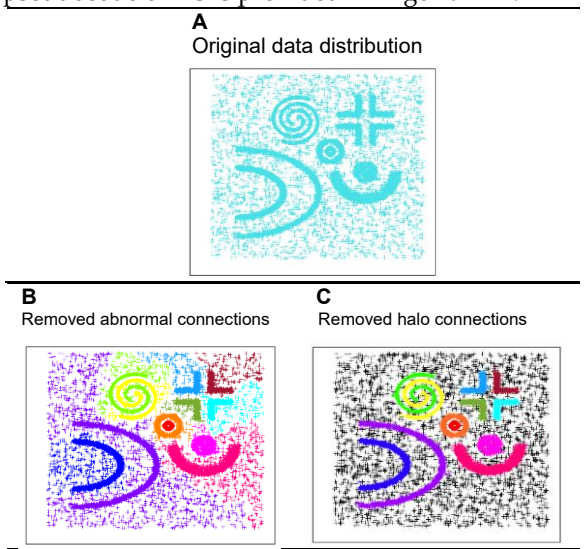


Figure 3. TC on the synthetic data set with 30% uniform noise. **(A)** is the original data distribution, including convex and non-convex clusters with 30% uniform noise; **(B)** illustrates the results of TC after removing the abnormal connections; **(C)** illustrates the results of TC after removing the halo connections, which is also the final partition.

2.6 Complexity Analysis

- **Time complexity**

According to the pseudocode of TC in Algorithm 1, if

the input is a distance matrix, the time complexity of Step 1 is $O(n^2)$. Steps 3 and 4 each require $O(n)$. Steps 6, 8 and 9 are in the loop. Suppose the loop needs to be executed m times, where $m \ll n$, the total time complexity of steps 6, 8 and 9 is $O(3mn)$, because each of them requires $O(n)$. Step 7 also needs to be executed m times, and its time complexity in each loop is $O(l^2)$ due to computing distances between neighboring clusters. However, initially, we regard each sample as its own cluster, so the distances between neighboring data samples can be regarded as the distances between neighboring clusters in the first loop without extra computing. Therefore, the total complexity of step 7 is $O\big((m-1)l^2\big)$. For step 10, since m loops generate a total of n-1 connections, its time complexity is $O(mn+n-1) = O\big((m+1)n-1\big)$. Steps 12, 14, 15 and 17 each require $O(n)$, and step 13 requires $O(nlogn)$. The time complexity of step 16 is $O(n+n-1-L)$, approximately equal to $O(2n)$. Similarly, step 18 also requires $O(2n)$. Hence, with a distance matrix as the input, TC's total time complexity approximately equal to $O(n^2)+O(nlogn)+O\big((4m+11)n\big)+O\big((m-1)l^2\big)$.

However, when using a fast approximate nearest neighbor method, we don't need to compute the distance matrix S, the time complexity of step 1 becomes 0 and $O(ml*logl)$ for step 7, giving a total time complexity of $O(nlogn)+O\big((4m+11)n\big)+O(ml*logl)$.

Space complexity

Over the entire algorithm, the following items need to be stored: the cluster set Γ with a mass of Θ, the sparse adjacency matrix for G, two properties of each connection M_{all} and D_{all}, the torque τ_i of each connection, and the torque gap $TGap_i$ between each connection in TSCL. Therefore, base space requirement is about $O(7n)$. This requirement does not change when using a fast approximate nearest neighbor method. However, if the input is a distance matrix, which needs to be stored additionally, the complexity increases to $O(n^2)+O(7n)$.

2.7 Algorithm Analysis

For better clarity, we visualized the step-by-step results of TC and the final results of several related methods on a synthetic data set, as shown in Fig. 4. TC follows the rule of Eq. (1) to gradually complete the merging of clusters, and finally automatically determines the exact number of clusters. In addition, we can see that in step 1, the red cluster and the blue cluster have been formed, which matches the ground truth. In steps 2-5, due to the constraint in Eq. (1), the red and blue clusters do not further merge their 1-nearest clusters but are "waiting" for other sub-clusters to complete the merging, where this process prevents wrong merging in conventional agglomerative methods.

On the other hand, according to Fig. 4, agglomerative clustering single-linkage (AC-S) is sensitive to outliers, leading to wrong results. Agglomerative clustering ward-linkage (AC-W) cannot detect clusters with complex shapes. FINCH [8] is an agglomerative method based on nearest neighbor statistics completely without any constraints. Obviously, the result of FINCH contains some wrong mergers. Density peak clustering (DPC) [39] is not robust to the varied density data sets, also leading to wrong results. Besides, all other methods need to manually set the number of clusters (or granularity levels) except for TC.

Algorithm 1: Algorithm of the proposed TC

1: Input: Distance matrix $S \in R^{n\times n}$ or data set $X \in R^{n\times D}$.
2: Output: Cluster partition $\phi = \{\zeta_1, \zeta_2, \dots, \zeta_{L+1}\}$ and cluster halo.
3: Initializing connected graph G.
4: Constructing cluster set $\Gamma = \{\zeta_i\}_{i=1}^{l}$ (Initially, regard each sample as a cluster, i.e., $l = n$).
5: while cluster set Γ have more than two clusters **do**
6 : Computing the mass θ_i of each cluster in Γ, where $\Theta = \{\theta_i\}_{i=1}^{l}$.
7: Searching the nearest cluster of ζ_i according to S or by using a fast approximate nearest neighbor method, e.g. kd-tree.
8: Generating the connections C_i and Updating G by Eq. (1).
9: Computing the two properties M_i and D_i of C_i by Eqs. (3)-(4), and save these to M_{all} and D_{all}, respectively.
10: Computing the connected components of G to update the cluster set Γ by Eq. (2).
11: end
12: Computing the torque τ_i of each connection based on M_{all} and D_{all} by Eq. (5).
13: Sorting all connections in descending order according to their corresponding torque values to get TSCL.
14: Computing $TGap_i$ between each consecutive connection in the TSCL by Eqs. (6)-(9).
15: Finding the largest $TGap_i$ denoted as $TGap_L$ and treat the L connections at the top of the TSCL as abnormal.
16: Updating G by removing the L abnormal connections, and then compute the connected components of G to obtain the final cluster partition $\phi = \{\zeta_1, \zeta_2, \dots, \zeta_{L+1}\}$.
17: Finding the halo connections by Eq. (10).
18: Updating G by removing the halo connections, and then compute the connected components of G to obtain the cluster halo.

3 Experiments and Results

To evaluate the performance of TC, we measured its performance on numerous synthetic and real-world data sets and compared its performance to other 19 well-known or latest algorithms. These algorithms include: K-means++ (K-M++) [15], GMM, Fuzzy clustering (Fuzzy) [40], Spectral clustering (SC) [41], [42], Hierarchical agglomerative clustering single-linkage (AC-S), complete-linkage (AC-C), average-linkage (AC-A), ward-linkage (AC-W), centroid-linkage (AC-CR) [37]; Density peak clustering (DPC) [39] and its three latest variants, Dynamic graph-based label propagation for density peak clustering (DPCLP) [22], Shared-nearest-neighbor-based density peak clustering

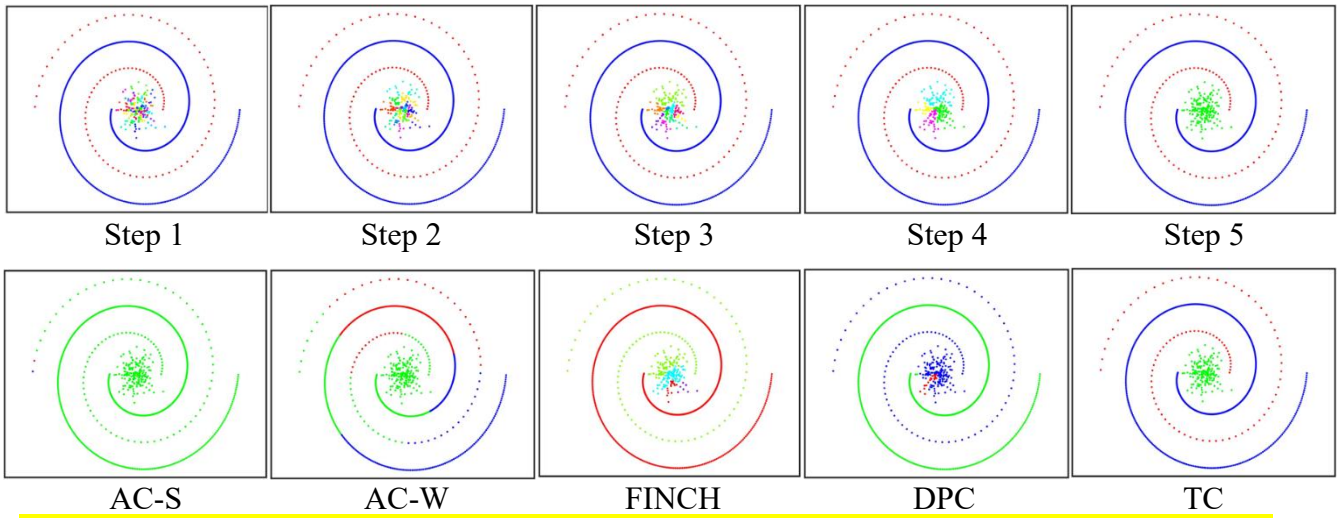


Figure 4. Visualization of the step-by-step results of TC (top) and the final results of related methods (bottom) on a synthetic data set.

(SNNDPC) [43], and Automatic density peak clustering (DPA) [44]; Efficient parameter-free clustering using first neighbor relations (FINCH) [8], DBSCAN (DB) [45], Mean-shift (MS) [46], Affinity Propagation (AP) [47], Border-Peeling clustering (BP), and Robust continuous clustering (RCC) [1]. Among them, DPA, FINCH, DB, MS, AP, BP, and RCC can automatically determine the number of clusters.

For each experiment, the free parameters of all the compared methods were set according to their best performance over a large range of possible configurations or runs, which gives a huge advantage to the compared methods. Implementation details on each of these baselines are provided in the supplement. Contrarily, the reported performance of TC is from just a single run.

All experiments were evaluated in terms of the two commonly-used external indices: normalized mutual information (NMI) [48] and accuracy (ACC). Additionally, we also compared TC with other automatic clustering algorithms for their ability to determine the optimal number of clusters. We counted the number of data sets each automatic clustering algorithm returns the exact ground-truth number of clusters, denoted as NGC.

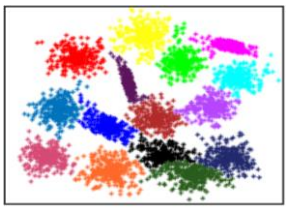
3.1 Evaluation on Nine Synthetic Data Sets

Fig. 5 presents the results of tests with nine different synthetic data sets reflecting seven challenges commonly faced in clustering. These data sets have been widely used as benchmark comparisons for many clustering algorithms [49]. Table 2 provide the descriptive statistics of these data sets.

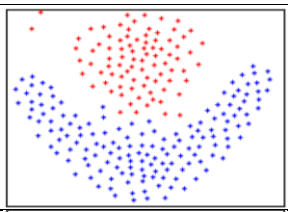
As the tests in Figs. 5A-5I show, the TC algorithm conquered every trial. In addition, TC automatically found the exact number of clusters for all nine data sets, matching the ground-truth numbers perfectly. As a means of visual comparison, we also conducted these same tests with K-means [11]. As shown in Fig S1 in the supplement, K-means failed on eight of the nine data sets, the exception being the first. Additionally, see Table 3 for the full quantitative comparison with the 19 state-of-the-art clustering algorithms on these synthetic data sets.

Figure 5. Results with seven different clustering challenges. As the results show, the proposed TC algorithm recognized all the clusters regardless of their shape, size, or density.

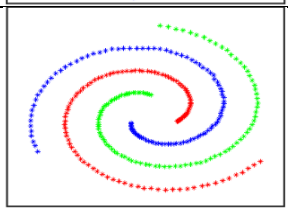
A.[50] **Highly overlapping:** TC was easily able to recognize the 15 clusters in this data set with substantial overlaps.



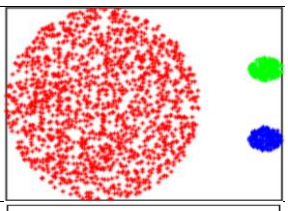
B.[51] **FLAME:** TC was able to find the two clusters in this case designed to test fuzzy clustering by local approximation of membership (FLAME).



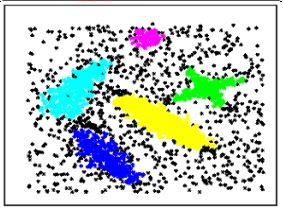
C.[52] **Spectral-path:** This data set was used to illustrate the performance of a path-based spectral clustering algorithm. TC was perfectly able to identify the three clusters without the need to generate a connectivity graph.



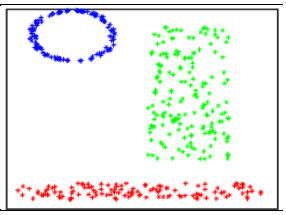
D.[53] **Unbalanced:** Severe imbalances in the data did not present a problem to TC as the hugely disproportionate clusters to the right show.



E.[39] **Noisy:** This data was originally used to showcase how a density-based clustering algorithm (fast search and finding peaks in density) handles noise. TC was able to detect the five clusters with lots of noise.

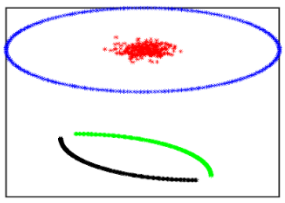


F.[54] **Heterogeneous geometric:** TC intuitively found the three clusters without the need to calculate point symmetry distances, as was required in.

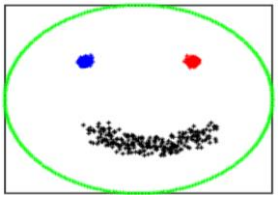


Multi-objective: Figs. 5G-5I show examples of multi-objective clustering. With these types of tasks, more than one type of clustering algorithm is needed to reveal all the different types of cluster structures in the data [55]. The current standard is to use ensemble learning to optimize multiple objective functions. TC was able to identify the different structures naturally.

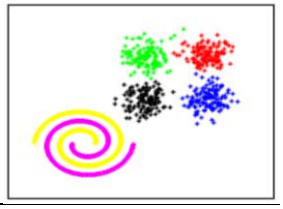
G. [56]



H. [57]



I. [58]



3.2 Evaluation on 11 Real-World Data Sets

In this section, we evaluated TC on a further 11 real-world clustering tasks (data sets) across five different domains: image recognition, biology, medicine, physics, and astronomy. Full descriptive statistics for the 11 data sets used are provided in Table 2. Here, we report the details of the tasks and TC's individual performance with each of them. Furthermore, Table 3 gives the full quantitative comparison with the 19 state-of-the-art clustering algorithms on these real-world data sets.

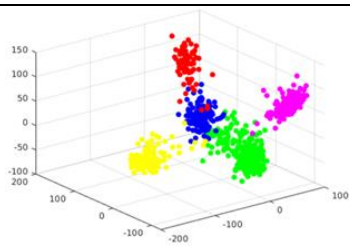
The image recognition experiments comprised handwritten digit recognition, face recognition, object recognition, and Pose, Illumination, Expression (PIE) recognition as four independent tasks. We used the popular benchmark data set MNIST [59] for the digit recognition task. Each digit from 0 to 9 should be clustered together to form a total of 10 clusters. TC correctly classified the digits with an accuracy of 99.22%.

The face recognition task was performed on the Olivetti Face Database (OFD) [60]. The results are shown in Fig. S2 color-coded by cluster. For ease of reporting, we have only included the first 100 images, denoted as OFD-F100. Using the similarity measure outlined in Ref. [61], TC completed the task with 92% accuracy.

The object recognition was conducted on the extremely high-dimensional data set COIL-100 (the Columbia University Image Library) [62] comprising 72 viewpoints on 100 objects, making a total of 7,200 samples and 49,152 pixels. TC completed the task with an NMI of 97.2%.

The PIE recognition was conducted on the CMU Pose, Illumination, and Expression (CMU-PIE) data set [63], which contains 13 different poses, under 43 different illumination conditions, and with 4 different expressions. TC completed the task with a perfect accuracy of 100%.

Figure 6. Projection of the five clusters (tumors) of the RNA-seq data set found by TC in a three-dimensional subspace.



The biology experiments comprised tasks in gene expression analysis, cell tracking analysis, and animal recognition. The gene expression task [64] was conducted with the RNA-seq data set [65], which is a random extraction of gene expressions in patients with five different types of tumors. TC correctly recognized all five with 99.88% accuracy. A more intuitive representation appears in Fig. 6 with a plot of the 20,531-dimensional feature space distilled to 3D space using PCA [66].

For the cell tracking task, we used the cancer cell tracking (Cell-track) data set from GitHub [67]. The goal is to use cell movements to determine whether they are in the RGDS or FSL layer. TC correctly tracked the cells to each of the two layers with 87.5% accuracy.

The last task, animal recognition, with the Zoo data set from the UCI Machine Learning Repository [68] was completed by TC with an accuracy of 92.08%.

The three tasks in the medicine domain were soybean disease diagnosis, and diagnosis of survival time in breast cancer patients.

The Soybean data set [69] contains observations of four diseases present in soybean plants. TC identified the diseases with a perfect accuracy of 100%.

Last in this domain was the survival time for breast cancer patients task using Haberman's Survival (Haberman) data set [70]. Survival times in this data are categorized into two groups – less or more than 5 years – based on three surgical features: the age of the patient at the time of their operation, the year of the operation, and the number of positive axillary nodes detected. TC identified three clusters with comparatively high accuracy, which is very close to the ground-truth of two clusters. Closer inspection of the additional cluster revealed samples with a reasonable survival time prediction in between the two bipartite clusters.

In the physics domain, we performed one task with the Atom data set [71], which contains 3D data similar to an atom kernel and hull. The data set contains two clusters in R^3 with a completely overlapping convex hull and has 800 samples in total [72]. This task is to discriminate between the kernel and hull of the atom. As shown in Fig. 7, this was another task TC performed with perfect accuracy.

In the astronomy domain, we tested the NASA Shuttle data set, which contains 58,000 multivariate measurements caused by seven different conditions in the radiator subsystem [1]. TC completed this task with an accuracy of 90.63%.

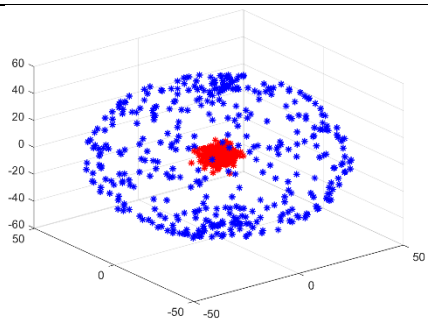


Figure 7. The result for the Atom data set. TC correctly identified the two clusters: the atom's kernel and hull.

Table 2. Statistics of the 20 data sets. The Noisy data set has no ground truth labels.

Data sets	Instances	Dimensions	Clusters	Imbalance
Highly overlapping	5000	2	15	~1
FLAME	240	2	2	~2
Spectral-path	312	2	3	~1
Unbalanced	2000	2	3	8
Noisy	4000	2	5	-
Heterogeneous geometric	400	2	3	~2
Multi-objective 1	1000	2	4	1
Multi-objective 2	1000	2	4	1
Multi-objective 3	1500	2	6	~4
OFD-F100	100	10304	10	1
MNIST	10000	4096	10	~1
COIL-100	7200	49152	100	1
Shuttle	58000	9	7	4558
RNA-seq	801	20531	5	~4
Haberman	306	3	2	~3
Zoo	101	16	7	~10
Atom	800	3	2	1
Soybean	47	35	4	~2
Cell-track	40	40	2	1
CMU-PIE	2856	1024	68	1

3.3 Result Analysis

3.3.1 Performance Advantage

According to the full quantitative comparison (see Table 3), TC indisputably outperformed all other state-of-the-art algorithms, given its best-in-show performance on 15 of the data sets.

In terms of rank (see Table 3), the next-best algorithm, SNNDPC, was about four times lower than TC in its ranking scores and, even then, needs to set the ground truth number of clusters in advance.

Moreover, beyond simple ranking metrics, there are some important quality-of-life issues to note:

- Most algorithms that outperformed TC on several data sets are sensitive to initialization/parameters, so the reported accuracies are the highest of many runs with different initializations or with different parameters. TC, however, is parameter-free and does not need any initialization, so the accuracy levels reported are from just a single run.
- Most algorithms require the analyst to specify the ground truth number of clusters before running the clustering procedure, whereas TC can automatically determine the number of clusters.

3.3.2 Automatic Determination of Number of Clusters

For a completely unsupervised clustering algorithm, it is important to be able to automatically determine the number of clusters. The TC algorithm, applied to the 20 data sets above, returned the exact or close to the exact number of clusters across the board without human intervention (15 exact, 5 close to). Table 4 highlights TC's performance against the seven comparators that can also automatically determine the number of clusters. TC was 50% more accurate than the next-best algorithm, DB, which only identified the correct number of clusters on 10 data sets even parameters tuning to get these results required many runs. The others ranged from 0 to 6 sets.

It is worth pointing out, however, that in many fields, choosing the right number of clusters can be subjective, depending more on the user's requirements than a ground-truth. This is why many clustering algorithms, including the well-known clustering method [39], DPC, all adopt similar kinds of decision graphs to visualize the cluster structure of the data set. Visualization helps users to estimate the ideal number of clusters, and offering a choice helps users better meet their own needs. However, because there is no objective and agreed definition of a cluster, people use different, subjective determinations of where the borders between clusters are [73]. Balcan et al. conducted valuable research on the problem of finding ground-truth clustering, which shows that using a list of partitions or a hierarchy instead of a single flat partition should be preferred [74]. Decision graphs do not identify a specific number of clusters, but they do provide a customized way for users to choose for themselves, which complies with the above views.

The decision graphs of the first 100 images of the Olivetti Face Database (see Fig. S4A) and the nine data sets in Figs. 4A-I (see Fig. S3) clearly show the cluster structure of each data set as compared to the decision graph for the DPC algorithm (see Fig. S4B). Further, the ease of estimating the ideal number of clusters with TC is clear. But these decision graphs provide another benefit. On the Olivetti Face Database data set, the TC algorithm identified 11 abnormal connections and removed them to leave 12. Yet, when looking at the decision graph, the actual number of abnormal connections is obviously nine. Removing these leaves 10 clusters, which is exactly the ground-truth number. The final recognition accuracy therefore moves from 92% with TC up to 95%. Thus, the decision graph is also helpful for manually correcting the TC algorithm in situations of excessive sensitivity.

Further, in the cases where the ground-truth number of clusters K is known, then K-1 connections with the largest τ_i in Eq. (5) can simply be regarded as abnormal connections. For example, with the COIL-100 data set, removing the 99 connections with the largest τ_i generates 100 clusters and 89.51% accuracy – higher than TC's at 86.33%. What this demonstrates is that if the ground-truth number of clusters is known in advance, TC's performance further improves.

3.4 Runtime

To more intuitively reflect the efficiency of TC, we compared the execution time of TC with that of all 19 algorithms. In general, three attributes of the data set, i.e., the number of samples, dimensions, and clusters, all affect the execution time. Therefore, we chose COIL-100 as the test

Table 3A. Performance comparison of all algorithms on all data sets, measured by NMI. The Noisy data set does not contain ground-truth labels, we removed it in the comparison. "NA" means not applicable.

Data sets/Methods	K-M++	GMM	Fuzzy	SC	AC-A	AC-W	AC-S	AC-C	AC-CR	DPC	DPCLP	SNNDPC	DPA	FINCH	DB	MS	AP	BP	RCC	TC
Highly overlapping	.9652	.9713	.9537	.0128	.9596	.9354	.0346	.8867	.9455	**.9747**	.8497	.9572	.9705	.8665	.2830	.9474	.7542	.8460	.7887	.9568
FLAME	.4843	.4477	.4422	.0479	.4832	.3297	.0479	.0770	.0479	.4132	.7937	.8288	.5805	.4896	.8374	.8673	.4408	.9083	.6492	**1.0000**
Spectral-path	.0012	.0678	.0003	1.0000	.0031	.0068	1.0000	.0106	.0119	1.0000	.3037	1.0000	.3903	.5359	1.0000	.3999	.5546	.2056	.5940	**1.0000**
Unbalanced	.4453	1.0000	.4429	1.0000	.6108	.6109	1.0000	.4406	.6351	1.0000	.6228	1.0000	.6526	.3720	1.0000	.6962	.3841	.5134	.3310	**1.0000**
Heterogeneous geometric	.8089	1.0000	.8016	1.0000	1.0000	1.0000	1.0000	.4611	1.0000	.7445	1.0000	1.0000	1.0000	.5583	1.0000	.8143	.5855	.8017	.5648	**1.0000**
Multi-objective 1	.8357	.9696	.6008	1.0000	.6895	.5981	.8633	.7008	.6636	.8044	.9673	1.0000	.8766	.6995	.9977	.7071	.6180	.8750	1.0000	**1.0000**
Multi-objective 2	.6807	.9448	.6072	1.0000	.6894	.6130	1.0000	.6920	.6775	.6663	1.0000	1.0000	.8660	.6846	1.0000	.7968	.6517	.7999	.8781	**1.0000**
Multi-objective 3	.7065	.8272	.5319	.7881	.7020	.7229	.8341	.7052	.7204	**.9950**	.6868	.7304	.7030	.7947	.9709	.8253	.5988	.9092	.5956	.9925
OFD-F100	.9057	.8225	.4923	.8218	.7278	.7927	.5846	.6063	.6744	.8666	.5195	.9136	.8538	.8746	.5195	.8166	.8379	.0000	.0000	**.9362**
MNIST	.9741	.8396	.3338	.9761	.9751	.9714	.0143	.9741	.9754	.9751	.0000	.9765	.8555	.9755	.9686	.8543	.7364	.7888	.7774	**.9767**
COIL-100	.8281	NA	.2866	.8564	.7256	.8353	.6990	.7428	.6577	.8657	.0760	.6919	.8504	.7897	.7223	.0270	.8388	.5510	.9628	**.9720**
Shuttle	.3247	.3888	.2415	.5860	.0277	.2671	.0385	.0516	.0285	.5769	NA	NA	.3700	.0368	.5113	.6060	NA	.3273	.4858	**.6389**
RNA-seq	.9808	.8400	.5711	.9948	.0464	.9860	.0318	.7252	.0320	.8348	.0000	.9895	.9745	.8785	.5712	.5716	.6633	.8960	.9287	**.9948**
Haberman	.0785	.0706	.0000	.0057	.0006	.0204	.0387	.0006	.0083	.0114	.0118	.0000	.0344	.0295	.0448	.0736	.0653	.0249	.0438	**.0847**
Zoo	NA	.8587	NA	.8438	.8728	.8640	.5025	**.9023**	.8916	.4937	.3530	.8082	.5257	.8725	.8605	NA	.7928	.5643	.8518	.8988
Atom	.3060	.9681	.3082	.0158	.2257	.2257	1.0000	.2107	.0620	.2735	1.0000	1.0000	.8236	.5560	1.0000	.6249	.4345	.6382	.4492	**1.0000**
Soybean	NA	.9441	NA	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	.8025	.0000	.7928	1.0000	.7539	.8489	NA	.7914	.6179	1.0000	**1.0000**
Cell-track	.4835	**.5617**	.3793	.3351	.0620	.5466	.0620	.0620	.0620	.3464	.0631	.3988	.3744	.3986	.3793	.3474	.3817	.1281	.3857	.4592
CMU-PIE	.5550	.4284	.1297	.8123	.5060	.5982	.9897	.4920	.3859	.9532	.3857	.3930	.9152	.7977	.7589	.0205	.8365	.2099	.3411	**1.0000**
Rank	10.5	7.9	16.2	7.3	11.7	10.3	9.5	12.9	12.4	8.7	12.7	6.7	7.7	10.5	7.1	11.1	12.9	12.1	10.4	**1.6**

Table 3B. Performance comparison of all algorithms on all data sets, measured by ACC.

Data sets/Methods	K-M++	GMM	Fuzzy	SC	AC-A	AC-W	AC-S	AC-C	AC-CR	DPC	DPCLP	SNNDPC	DPA	FINCH	DB	MS	AP	BP	RCC	TC
Highly overlapping	.9808	.9842	.9561	.0773	.9754	.9570	.0738	.8514	.9138	**.9858**	.8548	.9740	.9842	.7468	.1334	.9682	.3124	.7318	.4734	.9714
FLAME	.8583	.8417	.8500	.6458	.8333	.7208	.6458	.5167	.6458	.7875	.9625	.9708	.5375	.2292	.9375	.9792	.1375	.9833	.6167	**1.0000**
Spectral-path	.3526	.4295	.3401	1.0000	.3590	.3750	1.0000	.3878	.4038	1.0000	.5769	1.0000	.4744	.1571	1.0000	.3462	.1571	.5032	.2692	**1.0000**
Unbalanced	.6150	1.0000	.4732	1.0000	.5435	.5440	1.0000	.5320	.6795	1.0000	.7215	1.0000	.5160	.1015	1.0000	.7445	.0670	.3510	.1150	**1.0000**
Heterogeneous geometric	.9325	1.0000	.9325	1.0000	1.0000	1.0000	1.0000	.6075	1.0000	.8500	1.0000	1.0000	1.0000	.1350	1.0000	.8225	.1675	.7700	.1700	**1.0000**
Multi-objective 1	.8530	.9900	.6055	1.0000	.5890	.6580	.7490	.6180	.5760	.6560	.9890	1.0000	.7450	.4280	.9990	.5880	.1320	.8620	1.0000	**1.0000**
Multi-objective 2	.7910	.9820	.6860	1.0000	.8070	.7140	1.0000	.8090	.7830	.7320	1.0000	1.0000	.7500	.5490	1.0000	.7750	.2280	.7100	.8660	**1.0000**
Multi-objective 3	.7280	.7607	.5179	.5072	.7153	.7840	.7520	.7093	.7833	**.9987**	.6547	.7213	.5860	.6133	.9833	.6907	.1160	.9193	.1460	.9980
OFD-F100	.9100	.8200	.2990	.7593	.5900	.7600	.3300	.5200	.4400	.7800	.4100	.8200	.7700	.7700	.3700	.5900	.7000	.1000	.1000	**.9200**
MNIST	.9915	.7842	.2086	.9921	.9917	.9903	.1141	.9912	.9919	.9916	.1135	.9922	.6925	.9918	.9838	.8918	.2662	.6625	.5497	**.9922**
COIL-100	.5992	NA	.0233	.6368	.2794	.6053	.3504	.3521	.2175	.5482	.0187	.3089	.5385	.3922	.4313	.0107	.3529	.2532	.8307	**.8633**
Shuttle	.5456	.5116	.3000	.7581	.7834	.7663	.7862	.6386	.7837	.8893	NA	NA	.2912	.5788	.8232	.8855	NA	.0669	.6619	**.9063**
RNA-seq	.9950	.8939	.5546	.9988	.3645	.9963	.3758	.7403	.3758	.7990	.3745	.9975	.9850	.8240	.6067	.6242	.2784	.9288	.9001	**.9988**
Haberman	**.7582**	.6667	.5098	.5229	.7320	.7288	.7386	.7320	.7353	.5425	.7353	.6863	.6209	.4771	.7386	.7418	.1242	.7386	.3301	.7549
Zoo	NA	.8812	NA	.7228	.8614	.8812	.5248	.9109	.8911	.4752	.4752	.8515	.4950	.8713	.8317	NA	.6733	.5941	.8020	**.9208**
Atom	.7212	.9962	.7362	.5025	.6575	.6575	1.0000	.6450	.5250	.6963	1.0000	1.0000	.8163	.5675	1.0000	.6300	.0975	.6250	.2863	**1.0000**
Soybean	NA	.9787	NA	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	.8085	.3617	.7021	1.0000	.5106	.7872	NA	.5106	.5745	1.0000	**1.0000**
Cell-track	.8750	**.9000**	.8250	.8000	.5250	.9000	.5250	.5250	.5250	.8250	.6250	.8500	.7750	.7750	.8250	.5250	.6750	.5750	.6000	.8750
CMU-PIE	.2479	.2255	.0516	.6524	.1961	.2272	.9496	.1719	.1457	.7892	.1089	.1499	.7539	.3592	.6933	.0259	.3631	.0525	.0882	**1.0000**
Rank	8.9	7.7	14.8	7.4	10.2	8.0	8.5	11.5	10.4	7.8	11.3	6.1	9.7	13.6	6.6	12.6	16.6	12.6	13.1	**1.5**

Table 4. The performance comparison of eight automatic methods on predicting the ground-truth number of clusters. #C means the ground-truth number of clusters. NGC indicates the number of data sets each automatic clustering algorithm returns the exact ground-truth number of clusters.

Data sets/Methods	#C	DPA	FINCH	DB	MS	AP	BP	RCC	TC
Highly overlapping	**15**	**15**	56	2	**15**	116	29	10	**15**
FLAME	**2**	6	19	**2**	**2**	21	**2**	5	**2**
Spectral-path	**3**	5	56	**3**	6	35	**3**	33	**3**
Unbalanced	**3**	4	128	**3**	**3**	80	12	744	**3**
Noisy	**5**	4	29	**5**	**5**	144	54	176	**5**
Heterogeneous geometric	**3**	**3**	36	**3**	**3**	24	7	33	**3**
Multi-objective 1	**4**	6	26	**4**	2	40	6	**4**	**4**
Multi-objective 2	**4**	2	63	**4**	**4**	37	8	14	**4**
Multi-objective 3	**6**	10	7	7	5	75	**6**	104	**6**
OFD-F100	**10**	11	8	3	13	17	1	1	12
MNIST	**10**	42	**10**	**10**	15	80	20	149	**10**
COIL-100	**100**	129	44	64	11	568	32	90	129
Shuttle	**7**	336	16	8	56	-	204	144	4
RNA-seq	**5**	6	4	3	63	36	**5**	4	**5**
Haberman	**2**	5	3	**2**	5	19	1	6	3
Zoo	**7**	12	8	3	-	11	3	14	6
Atom	**2**	**2**	29	**2**	13	55	9	80	**2**
Soybean	**4**	**4**	12	3	-	9	2	**4**	**4**
Cell-track	**2**	3	4	1	10	9	1	11	**2**
CMU-PIE	**68**	61	271	67	4	231	3	6	**68**
NGC	-	4	1	10	6	0	4	2	**15**

Table 5. Runtime comparison of TC with the state-of-art clustering algorithms on COIL-100 data set.

K-M++	Fuzzy	GMM	SC	AC-A	AC-W	AC-S	AC-C	AC-CR	DPC
0:00:28	00:12:16	NA	00:03:41	00:02:02	00:02:27	00:01:53	00:01:57	00:02:07	00:02:11
DPCLP	SNNDPC	DPA	FINCH	DB	MS	AP	BP	RCC	TC
00:07:25	00:15:08	00:02:28	00:00:48	00:04:03	00:07:09	00:07:59	11:47:26	06:16:51	00:00:31

data set because all three attributes are reflected in relatively large numbers. All algorithms were implemented in Matlab or Python. All of the tests were run on a workstation with two 14-core Intel Xeon 6132 CPUs running at 2.6 GHz and 3.7 GHz, as well a s 96GB of RAM. The average execution time is reported for the clustering algorithms that need to be executed multiple times. The author-provided code for GMM breaks on the test data set. The results in Table 5 show the running time of TC is less than that of all other algorithms except K-means++.

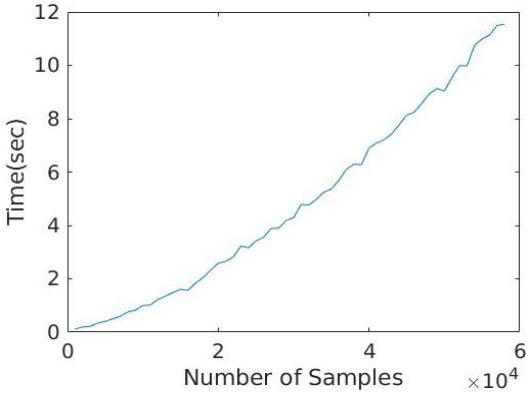


Figure 8. Runtime of TC. Runtime is evaluated as a function of data set size, using randomly sampled subsets of different sizes from the Shuttle data set.

Additionally, Fig. 8 shows the running time of TC on randomly sampled subsets of the Shuttle data set. The proposed TC can produce clustering results in roughly 12 seconds for all subsets.

3.5 Further Evaluation on 56 Data Sets with Peculiar Characteristics

In this section, we conducted an additional comprehensive evaluation on 56 data sets in total that includes 27 data sets with noise, outliers, overlaps, or other peculiar distributions, nine synthetic data sets with unbalanced clusters,

five synthetic data sets with uniform noise and 15 poorly-

Data sets	CMU-PIE	COIL-100	COIL-40	COIL-20	FRGC-v2.0	UMIST	Pendigits
Rank 1	1 JULE [75] 2016	**.985** JULE [75] 2016	.967 A-DSSC [76] 2020	**1** JULE [75] 2016	**.651** DNB [77] 2021	.917 DSC-FEDL [78] 2020	**.868** EAEDC [79] 2021
Rank 2	1 DDSNnet [80] 2021	.946 A-DSSC [76] 2020	.963 J-DSSC [76] 2020	.981 DSC-FEDL [78] 2020	.610 DEPICT [81] 2017	.893 S^2DSCAG [82] 2020	.863 N2D [83] 2019
Rank 3	.970 DAutoED [84] 2021	.943 J-DSSC [76] 2020	.951 DSC-FEDL [78] 2020	.979 SADSC [85] 2021	.580 MI-ADM [86] 2021	.890 DSC-DAG [82] 2020	.820 DnC-SC[87] 2021
Rank 4	.965 MI-ADM [86] 2021	.910 DGMM [88] 2021	.928 RGRL [89] 2020	.974 S^2DSCAG [82] 2020	.574 JULE [75] 2016	.881 RGRL [89] 2020	.817 DipDECK [90] 2021
Rank 5	.964 DEPICT [81] 2017	.905 DBC [91] 2018	.920 DASC [92] 2018	.958 DSC-DAG [82] 2020	.544 DPSC [93] 2021	.877 JULE [75] 2016	.814 GCML [94] 2022
Rank 6	.925 DPSC [93] 2021	.886 DDSNnet[80] 2021	.916 DSC-DAG [82] 2020	.910 DGMM [88] 2021	.522 DDSNnet [80] 2021	.851 DNB [77] 2021	.801 AESC[95] 2020
TC	**1**	.972	**.989**	.960	.586	**.931**	.849

Table. 6. Comparison to Deep Clustering algorithms, measured by NMI.

separated, high-dimensional gene expression data sets, collected for the purpose of evaluating clustering algorithms. The results are reported in Tables S5-S8 and Figs S5-S7. Overall, TC still retained a great performance advantage on these data sets.

3.6 Comparison to Deep Clustering algorithms on Challenging Image Data Sets

Image data sets usually have very high dimensions, and traditional clustering algorithms often cannot achieve good results on them. In recent years, a lot of works focused on using deep neural networks to learn a clustering-friendly low dimensional representation, resulting in a significant increase of clustering performance on image data sets [96]. Hence, in this section, we also compared TC with the latest state-of-the-art deep clustering algorithms on several challenging image data sets, including UMIST [97], FRGC-v2.0 [98], COIL-20 [99], COIL-40 [62], Pendigits [100], and the two mentioned above, COIL-100 [62] and CMU-PIE [63]. We run TC directly on the raw pixels (or features) of these image data sets without any other representation. Further, we combine "papers with code", a website that ranks the performance of open-source algorithms, and the latest papers on deep clustering algorithms, to list the results of the top six deep clustering algorithms that perform best on these seven image data sets (measured by NMI), as shown in Table 6. On CMU-PIE, COIL-40, and UMIST data sets, TC outperforms all state-of-the-art deep clustering algorithms. Besides, the results of TC are also competitive on other data sets.

In summary, TC without any deep representation can achieve better or close performance on challenging image data sets, compared with state-of-the-art deep clustering algorithms. Deep clustering algorithms also face some challenges. For example, they have several hyper-parameters that are non-trivial to set, lack interpretability, and have high computational complexity.

4 Discussion

4.1 Differences between TC and Other Hierarchical Clustering Algorithms

Although TC appears to be a hierarchy-based clustering algorithm, it is different from the existing algorithms in several major ways.

First, most of the previous hierarchical clustering algorithms are completely based on the nearest neighbors' statistics without constraints. However, we introduce a simple but very effective constraint (i.e., the requirement in Eq. (1)) in TC, which prevents wrong merging usefully (see Fig. 4). This idea is inspired by the gravitational interactions of galaxy minor mergers. Second, in each step of TC, if any two neighboring clusters satisfy the requirement of Eq. (1), a connection can be formed, and thus mergers can be performed in parallel. That means a large cluster can form within very few steps (see Table S3), greatly improving clustering efficiency and reducing the algorithm's execution time (Fig. 9). However, standard hierarchical clustering algorithms need to perform merging at least n-K times to get K clusters. Third, TC algorithm automatically determines the number of clusters by removing abnormal connections based on a new *TGap* metric. Instead, most of the existing hierarchical algorithms still need to manually set the number of clusters (or granularity levels), even if dendrograms are provided. Finally, TC is robust to noise and outliers and can identify noise clusters. However, many of the classic agglomerative clustering algorithms are not robust to noise [101]. As the case study in Fig. 4 and the empirical results in Table 3, TC outperforms other hierarchy-based clustering algorithms.

4.2 Differences between TC and Density Peak Clustering Techniques

Even though the decision graph of TC is similar to that of

DPC techniques, it is also different from the existing algorithms in several major ways.

First, decision objects are different. TC determines which connections between neighboring clusters are abnormal connections, and then prunes the clustering tree by removing them to get the results. The DPC is to decide which data samples are cluster centers, and then complete the label assignments of the remaining samples according to them. Excluding the distance between clusters (or between samples), as the basis for determining abnormal connections, TC only needs to count the number of samples (i.e., mass) contained in each cluster to obtain M_i in Eq. (3), without any hyper-parameters. While DPC uses some density estimators to estimate the density of each sample to determine the cluster centers, where these density estimators usually contain hyper-parameters such as cutoff distance. Second, label assignment strategies are different. TC leverages the constrained method of merging in Eq. (1) to assign labels to samples in clusters, which can effectively improve accuracy. DPC, on the other hand, assigns labels to samples based on selected cluster centers (i.e., density peaks). However, once the cluster centers are wrongly chosen, then there may be many more samples subsequently misassigned. For example, in the varied density data set of Fig. 4, DPC erroneously selects two cluster centers (i.e., the blue and red ones) in a ground truth cluster, which eventually leads to wrong label assignment. Finally, robustness is different. DPC is not robust to the varied density data sets, since it assumes that density peaks must be cluster centers. While TC has parallelism in the merging process, that is, if any two neighboring clusters satisfy the requirement of Eq. (1), a connection can be formed, which can effectively reduce the sensitivity to varied density in conventional hierarchical clustering (see Fig. 4).

4.3 Potential limitations of TC

On the one hand, the merging process of TC relies on nearest-neighbor statistics and leverages a method like single-linkage to measure the distance between clusters. Therefore, the performance of TC on some high-dimensional and sparse data sets is not particularly satisfactory (see Table S8), even if still outperforming related methods. On the other hand, since TC relies on the global mean variable values (i.e., mean_M, mean_D, and mean_$\frac{D}{M}$) to detect noise clusters, it may not be able to accurately identify non-uniform noise. We will address these issues in future work.

5 Conclusion

Whether compared to the classic or very recent methods, TC demonstrates itself to be a highly accurate algorithm, superior in performance to all its counterparts and with an unprecedented level of universality. Overall, we have presented a clustering algorithm that is: parameter-free, can recognize all kinds of clusters regardless of their shape, size or density; does not depend on a priori knowledge; is robust to noise and outliers; does not need any initialization; automatically determines the number of clusters, and does not demand a manually-specified stopping condition. Moreover, the ability to use any desired method of 1-nearest-cluster computation means TC is scalable to large data sets with a relatively low computational overhead and a reasonable time complexity, especially when choosing fast approximate nearest neighbor methods, such as k-d tree or locality-sensitive hashing [7], [8]. In this article, we presented many test cases to showcase TC's versatility. Even more experimental details and comparisons of clustering quality versus state-of-the-art methods are provided in Tables S1-S7 and Figs. S5-S9 in the supplementary materials.

References

[1] S. A. Shah and V. Koltun, "Robust continuous clustering," *Proc. Natl. Acad. Sci.*, vol. 114, no. 37, pp. 9814–9819, Sep. 2017, doi: 10.1073/pnas.1700770114.

[2] J. Han, J. Pei, and M. Kamber, *Data Mining: Concepts and Techniques*. Elsevier, 2011.

[3] C. Fraley and A. E. Raftery, "How Many Clusters? Which Clustering Method? Answers Via Model-Based Cluster Analysis," *Comput. J.*, vol. 41, no. 8, pp. 578–588, Jan. 1998, doi: 10.1093/comjnl/41.8.578.

[4] A. Saxena *et al.*, "A review of clustering techniques and developments," *Neurocomputing*, vol. 267, pp. 664–681, Dec. 2017, doi: 10.1016/j.neucom.2017.06.053.

[5] L. McInnes and J. Healy, "Accelerated Hierarchical Density Based Clustering," in *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, Nov. 2017, pp. 33–42. doi: 10.1109/ICDMW.2017.12.

[6] M. Dash, H. Liu, P. Scheuermann, and K. L. Tan, "Fast hierarchical clustering and its validation," *Data Knowl. Eng.*, vol. 44, no. 1, pp. 109–138, Jan. 2003, doi: 10.1016/S0169-023X(02)00138-6.

[7] H. Koga, T. Ishibashi, and T. Watanabe, "Fast agglomerative hierarchical clustering algorithm using Locality-Sensitive Hashing," *Knowl. Inf. Syst.*, vol. 12, no. 1, pp. 25–53, May 2007, doi: 10.1007/s10115-006-0027-5.

[8] S. Sarfraz, V. Sharma, and R. Stiefelhagen, "Efficient Parameter-Free Clustering Using First Neighbor Relations," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019, pp. 8926–8935. doi: 10.1109/CVPR.2019.00914.

[9] S. Kpotufe and U. von Luxburg, "Pruning nearest neighbor cluster trees," *ArXiv11050540 Cs Stat*, May 2011, Accessed: Nov. 23, 2020. [Online]. Available: http://arxiv.org/abs/1105.0540

[10] K. Chaudhuri, S. Dasgupta, S. Kpotufe, and U. von Luxburg, "Consistent Procedures for Cluster Tree Estimation and Pruning," *IEEE Trans. Inf. Theory*, vol. 60, no. 12, pp. 7900–7912, Dec. 2014, doi: 10.1109/TIT.2014.2361055.

[11] J. MacQueen, "SOME METHODS FOR CLASSIFICATION AND ANALYSIS OF MULTIVARIATE OBSERVATIONS," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, 1967, vol. 1(14), pp. 281–297.

[12] D. Pelleg and A. Moore, "X-means: Extending K-means with Efficient Estimation of the Number of Clusters," in *In Proceedings of the 17th International Conf. on Machine Learning*, 2000, pp. 727–734.

[13] I. S. Dhillon, Y. Guan, and B. Kulis, "Kernel k-means: spectral

clustering and normalized cuts," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, New York, NY, USA, Aug. 2004, pp. 551–556. doi: 10.1145/1014052.1014118.

[14] Rong Zhang and A. I. Rudnicky, "A large scale clustering scheme for kernel K-Means," in *Object recognition supported by user interaction for service robots*, Aug. 2002, vol. 4, pp. 289–292 vol.4. doi: 10.1109/ICPR.2002.1047453.

[15] D. Arthur and S. Vassilvitskii, "k-means++: The Advantages of Careful Seeding," *Proc. Eighteenth Annu. ACM-SIAM Symp. Discrete Algorithms Soc. Ind. Appl. Math.*, pp. 1027–1035, 2007.

[16] J. Yang, Y. Ma, X. Zhang, S. Li, and Y. Zhang, "An Initialization Method Based on Hybrid Distance for *k* -Means Algorithm," *Neural Comput.*, vol. 29, no. 11, pp. 3094–3117, Nov. 2017, doi: 10.1162/neco_a_01014.

[17] M. Wang, W. Zuo, and Y. Wang, "An improved density peaks-based clustering method for social circle discovery in social networks," *Neurocomputing*, vol. 179, pp. 219–227, Feb. 2016, doi: 10.1016/j.neucom.2015.11.091.

[18] T. Liu, H. Li, and X. Zhao, "Clustering by Search in Descending Order and Automatic Find of Density Peaks," *IEEE Access*, vol. 7, pp. 133772–133780, 2019, doi: 10.1109/ACCESS.2019.2939437.

[19] Z. Liang and P. Chen, "Delta-density based clustering with a divide-and-conquer strategy: 3DC clustering," *Pattern Recognit. Lett.*, vol. 73, pp. 52–59, Apr. 2016, doi: 10.1016/j.patrec.2016.01.009.

[20] A. Lotfi, P. Moradi, and H. Beigy, "Density peaks clustering based on density backbone and fuzzy neighborhood," *Pattern Recognit.*, vol. 107, p. 107449, Nov. 2020, doi: 10.1016/j.patcog.2020.107449.

[21] R. J. G. B. Campello, D. Moulavi, and J. Sander, "Density-Based Clustering Based on Hierarchical Density Estimates," in *Advances in Knowledge Discovery and Data Mining*, Berlin, Heidelberg, 2013, pp. 160–172. doi: 10.1007/978-3-642-37456-2_14.

[22] S. A. Seyedi, A. Lotfi, P. Moradi, and N. N. Qader, "Dynamic graph-based label propagation for density peaks clustering," *Expert Syst. Appl.*, vol. 115, pp. 314–328, Jan. 2019, doi: 10.1016/j.eswa.2018.07.075.

[23] W. Zhang and J. Li, "Extended fast search clustering algorithm: widely density clusters, no density peaks," *Comput. Sci. Inf. Technol. CS IT*, pp. 01–17, Apr. 2015, doi: 10.5121/csit.2015.50701.

[24] J. Xie, H. Gao, W. Xie, X. Liu, and P. W. Grant, "Robust clustering by detecting density peaks and assigning points based on fuzzy weighted K-nearest neighbors," *Inf. Sci.*, vol. 354, pp. 19–40, Aug. 2016, doi: 10.1016/j.ins.2016.03.011.

[25] M. Du, S. Ding, and H. Jia, "Study on density peaks clustering based on k-nearest neighbors and principal component analysis," *Knowl.-Based Syst.*, vol. 99, pp. 135–145, May 2016, doi: 10.1016/j.knosys.2016.02.001.

[26] L. Scrucca and A. Raftery, "Improved initialisation of model-based clustering using Gaussian hierarchical partitions," *Adv. Data Anal. Classif.*, vol. 9, no. 4, pp. 447–460, 2015, Accessed: Nov. 23, 2020. [Online]. Available: https://ideas.repec.org/a/spr/advdac/v9y2015i4p447-460.html

[27] A. O'Hagan and A. White, "Improved model-based clustering performance using Bayesian initialization averaging," *Comput. Stat.*, vol. 34, no. 1, pp. 201–231, Mar. 2019, doi: 10.1007/s00180-018-0855-2.

[28] E. Kebriaei, K. Bijari, and H. Zare, "Improved model-based clustering using evolutionary optimization," in *2017 Artificial Intelligence and Robotics (IRANOPEN)*, Apr. 2017, pp. 182–187. doi: 10.1109/RIOS.2017.7956464.

[29] J. Chen, X. Lin, Q. Xuan, and Y. Xiang, "FGCH: a fast and grid based clustering algorithm for hybrid data stream," *Appl. Intell.*, vol. 49, no. 4, pp. 1228–1244, Apr. 2019, doi: 10.1007/s10489-018-1324-x.

[30] B. Wu and B. M. Wilamowski, "A Fast Density and Grid Based Clustering Method for Data With Arbitrary Shapes and Noise," *IEEE Trans. Ind. Inform.*, vol. 13, no. 4, pp. 1620–1628, Aug. 2017, doi: 10.1109/TII.2016.2628747.

[31] Q. Liu, K. Zhang, J. Shen, Z. Fu, and N. Linge, "GLRM: An improved grid-based load-balanced routing method for WSN with single controlled mobile sink," in *2016 18th International Conference on Advanced Communication Technology (ICACT)*, Jan. 2016, pp. 34–38. doi: 10.1109/ICACT.2016.7423264.

[32] G. De Lucia and J. Blaizot, "The hierarchical formation of the brightest cluster galaxies," *Mon. Not. R. Astron. Soc.*, vol. 375, no. 1, pp. 2–14, Feb. 2007, doi: 10.1111/j.1365-2966.2006.11287.x.

[33] M. S. Petersen, M. D. Weinberg, and N. Katz, "Using torque to understand barred galaxy models," *Mon. Not. R. Astron. Soc.*, vol. 490, no. 3, pp. 3616–3632, Dec. 2019, doi: 10.1093/mnras/stz2824.

[34] E. R. Stanway *et al.*, "Exploring the cosmic evolution of habitability with galaxy merger trees," *Mon. Not. R. Astron. Soc.*, vol. 475, no. 2, pp. 1829–1842, Apr. 2018, doi: 10.1093/mnras/stx3305.

[35] S. Cole, C. G. Lacey, C. M. Baugh, and C. S. Frenk, "Hierarchical galaxy formation," *Mon. Not. R. Astron. Soc.*, vol. 319, no. 1, pp. 168–204, Nov. 2000, doi: 10.1046/j.1365-8711.2000.03879.x.

[36] K. R. V. Casteels *et al.*, "Galaxy And Mass Assembly (GAMA): refining the local galaxy merger rate using morphological information," *Mon. Not. R. Astron. Soc.*, vol. 445, no. 2, pp. 1157–1169, Dec. 2014, doi: 10.1093/mnras/stu1799.

[37] S. C. Johnson, "Hierarchical clustering schemes," *Psychometrika*, vol. 32, no. 3, pp. 241–254, Sep. 1967, doi: 10.1007/BF02289588.

[38] Y. Jeon, J. Yoo, J. Lee, and S. Yoon, "NC-Link: A New Linkage Method for Efficient Hierarchical Clustering of Large-Scale Data," *IEEE Access*, vol. 5, pp. 5594–5608, 2017, doi: 10.1109/ACCESS.2017.2690987.

[39] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, Jun. 2014, doi: 10.1126/science.1242072.

[40] J. C. Bezdek, R. Ehrlich, and W. Full, "FCM: The fuzzy c-means clustering algorithm," *Comput. Geosci.*, vol. 10, no. 2–3, pp. 191–203, Jan. 1984, doi: 10.1016/0098-3004(84)90020-7.

[41] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On Spectral Clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems 14*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds. MIT Press, 2002, pp. 849–856. Accessed: Mar. 05, 2020. [Online]. Available: http://papers.nips.cc/paper/2092-

on-spectral-clustering-analysis-and-an-algorithm.pdf

[42] Jianbo Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000, doi: 10.1109/34.868688.

[43] R. Liu, H. Wang, and X. Yu, "Shared-nearest-neighbor-based clustering by fast search and find of density peaks," *Inf. Sci.*, vol. 450, pp. 200–226, 2018, doi: 10.1016/j.ins.2018.03.031.

[44] M. d'Errico, E. Facco, A. Laio, and A. Rodriguez, "Automatic topography of high-dimensional data sets by non-parametric density peak clustering," *Inf. Sci.*, vol. 560, pp. 476–492, 2021, doi: 10.1016/j.ins.2021.01.010.

[45] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "Density-based spatial clustering of applications with noise," 1996, vol. Vol. 240.

[46] K. Fukunaga and L. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE Trans. Inf. Theory*, vol. 21, no. 1, pp. 32–40, Jan. 1975, doi: 10.1109/TIT.1975.1055330.

[47] B. J. Frey and D. Dueck, "Clustering by Passing Messages Between Data Points," *Science*, vol. 315, no. 5814, pp. 972–976, Feb. 2007, doi: 10.1126/science.1136800.

[48] A. Strehl and J. Ghosh, "Cluster Ensembles --- A Knowledge Reuse Framework for Combining Multiple Partitions," *J. Mach. Learn. Res.*, vol. 3, no. Dec, pp. 583–617, 2002, Accessed: Mar. 20, 2020. [Online]. Available: http://www.jmlr.org/papers/v3/strehl02a.html

[49] B. Tomas, "Clustering benchmarks," 2019. https://github.com/deric/clustering-benchmark

[50] P. Fränti and O. Virmajoki, "Iterative shrinking method for clustering problems," *Pattern Recognit.*, vol. 39, no. 5, pp. 761–775, May 2006, doi: 10.1016/j.patcog.2005.09.012.

[51] L. Fu and E. Medico, "FLAME, a novel fuzzy clustering method for the analysis of DNA microarray data," *BMC Bioinformatics*, vol. 8, no. 1, p. 3, Jan. 2007, doi: 10.1186/1471-2105-8-3.

[52] H. Chang and D.-Y. Yeung, "Robust path-based spectral clustering," *Pattern Recognit.*, vol. 41, no. 1, pp. 191–203, Jan. 2008, doi: 10.1016/j.patcog.2007.04.010.

[53] S. Guha, R. Rastogi, and K. Shim, "Cure: an efficient clustering algorithm for large databases," *Inf. Syst.*, vol. 26, no. 1, pp. 35–58, Mar. 2001, doi: 10.1016/S0306-4379(01)00008-4.

[54] M.-C. Su, C.-H. Chou, and C.-C. Hsieh, "Fuzzy C-Means Algorithm with a Point Symmetry Distance," *Int. J. Fuzzy Syst.*, no. 7(4), pp. 175–181, 2005.

[55] M. H. C. Law, A. P. Topchy, and A. K. Jain, "Multiobjective data clustering," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, Jun. 2004, vol. 2, p. II–II. doi: 10.1109/CVPR.2004.1315194.

[56] A. Garcia-Piquer, A. Sancho-Asensio, A. Fornells, E. Golobardes, G. Corral, and F. Teixidó-Navarro, "Toward high performance solution retrieval in multiobjective clustering," *Inf. Sci.*, vol. 320, pp. 12–25, Nov. 2015, doi: 10.1016/j.ins.2015.04.041.

[57] H. Julia and K. Joshua, "Multiobjective clustering with automatic determination of the number of clusters," *Tech. Rep.*, 2004.

[58] K. Faceli, T. C. Sakata, M. C. P. de Souto, and A. C. P. L. F. de Carvalho, "Partitions selection strategy for set of clustering solutions," *Neurocomputing*, vol. 73, no. 16, pp. 2809–2819, Oct. 2010, doi: 10.1016/j.neucom.2010.03.028.

[59] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, no. 86(11), pp. 2278–2324, 1998.

[60] F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification," in *Proceedings of 1994 IEEE Workshop on Applications of Computer Vision*, Dec. 1994, pp. 138–142. doi: 10.1109/ACV.1994.341300.

[61] M. P. Sampat, Z. Wang, S. Gupta, A. C. Bovik, and M. K. Markey, "Complex Wavelet Structural Similarity: A New Image Similarity Index," *IEEE Trans. Image Process.*, vol. 18, no. 11, pp. 2385–2401, Nov. 2009, doi: 10.1109/TIP.2009.2025923.

[62] S. A. Nene, S. K. Nayar, and H. Murase, "Columbia Object Image Library (COIL-100)," *Tech. Rep.*, no. CUCS-006-96, 1996.

[63] T. Sim, S. Baker, and M. Bsat, "The CMU Pose, Illumination, and Expression (PIE) database," in *Proceedings of Fifth IEEE International Conference on Automatic Face Gesture Recognition*, May 2002, pp. 53–58. doi: 10.1109/AFGR.2002.1004130.

[64] V. Y. Kiselev, T. S. Andrews, and M. Hemberg, "Challenges in unsupervised clustering of single-cell RNA-seq data," *Nat. Rev. Genet.*, vol. 20, no. 5, pp. 273–282, May 2019, doi: 10.1038/s41576-018-0088-9.

[65] The Cancer Genome Atlas Research Network *et al.*, "The Cancer Genome Atlas Pan-Cancer analysis project," *Nat. Genet.*, vol. 45, no. 10, pp. 1113–1120, Oct. 2013, doi: 10.1038/ng.2764.

[66] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemom. Intell. Lab. Syst.*, vol. 2, no. 1, pp. 37–52, Aug. 1987, doi: 10.1016/0169-7439(87)80084-9.

[67] A. Dogan, "Cell-Tracking-Analysis-with-K-Means-Clustering-Method." https://github.com/ddaskan/Cell-Tracking-Analysis-with-K-Means-Clustering-Method

[68] "Data Sets - UCI Machine Learning." https://archive.ics.uci.edu/ml/datasets.php

[69] "Using Weighted Networks to Represent Classification Knowledge in Noisy Domains," *Mach. Learn. Proc. 1988*, pp. 121–134, Jan. 1988, doi: 10.1016/B978-0-934613-64-4.50018-9.

[70] S. J. Haberman, "Generalized residuals for log-linear models," 1976, pp. 104–122.

[71] A. Ultsch, "Strategies for an artificial life system to cluster high dimensional data," *Abstr. Synth. Princ. Living Syst.*, vol. GWAL-6, pp. 128–137, 2004.

[72] M. C. Thrun and A. Ultsch, "Clustering benchmark datasets exploiting the fundamental clustering problems," *Data Brief*, vol. 30, p. 105501, Jun. 2020, doi: 10.1016/j.dib.2020.105501.

[73] V. Estivill-Castro, "Why so many clustering algorithms: a position paper," *ACM SIGKDD Explor. Newsl.*, vol. 4, no. 1, pp. 65–75, Jun. 2002, doi: 10.1145/568574.568575.

[74] M.-F. Balcan, A. Blum, and S. Vempala, "A discriminative framework for clustering via similarity functions," in *Proceedings of the fortieth annual ACM symposium on Theory of computing*, Victoria, British Columbia, Canada, May 2008, pp. 671–680. doi: 10.1145/1374376.1374474.

[75] J. Yang, D. Parikh, and D. Batra, "Joint Unsupervised Learning of Deep Representations and Image Clusters," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 5147–5156. doi: 10.1109/CVPR.2016.556.

[76] D. Lim, R. Vidal, and B. Haeffele, "Doubly Stochastic Subspace Clustering," *ArXiv*, 2020.

[77] Z. Wang, Y. Ni, B. Jing, D. Wang, H. Zhang, and E. Xing, "DNB: A Joint Learning Framework for Deep Bayesian Nonparametric Clustering," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–11, 2021, doi: 10.1109/TNNLS.2021.3085891.

[78] Q. Huang, Y. Zhang, H. Peng, T. Dan, W. Weng, and H. Cai, "Deep subspace clustering to achieve jointly latent feature extraction and discriminative learning," *Neurocomputing*, vol. 404, pp. 340–350, 2020, doi: 10.1016/j.neucom.2020.04.120.

[79] X. Huang, Z. Hu, and L. Lin, "Deep clustering based on embedded auto-encoder," *Soft Comput.*, 2021, doi: 10.1007/s00500-021-05934-8.

[80] W. Wang, F. Chen, Y. Ge, S. Huang, X. Zhang, and D. Yang, "Discriminative deep semi-nonnegative matrix factorization network with similarity maximization for unsupervised feature learning," *Pattern Recognit. Lett.*, vol. 149, pp. 157–163, 2021, doi: 10.1016/j.patrec.2021.06.013.

[81] K. G. Dizaji, A. Herandi, C. Deng, W. Cai, and H. Huang, "Deep Clustering via Joint Convolutional Autoencoder Embedding and Relative Entropy Minimization," Oct. 2017, pp. 5747–5756. doi: 10.1109/ICCV.2017.612.

[82] Z. Yu, Z. Zhang, W. Cao, C. Liu, J. Philip Chen, and H. S. Wong, "GAN-based Enhanced Deep Subspace Clustering Networks," *IEEE Trans. Knowl. Data Eng.*, pp. 1–1, 2020, doi: 10.1109/TKDE.2020.3025301.

[83] R. McConville, R. Santos-Rodriguez, R. J. Piechocki, and I. Craddock, "N2D: (Not Too) Deep Clustering via Clustering the Local Manifold of an Autoencoded Embedding," *ArXiv E-Prints*, vol. 1908, p. arXiv:1908.05968, Aug. 2019, Accessed: Feb. 11, 2021. [Online]. Available: http://adsabs.harvard.edu/abs/2019arXiv190805968M

[84] M. Yang and S. Xu, "Orthogonal Nonnegative Matrix Factorization using a novel deep Autoencoder Network," *Knowl.-Based Syst.*, vol. 227, p. 107236, 2021, doi: 10.1016/j.knosys.2021.107236.

[85] Z. Chen, S. Ding, and H. Hou, "A novel self-attention deep subspace clustering," *Int. J. Mach. Learn. Cybern.*, vol. 12, no. 8, pp. 2377–2387, 2021, doi: 10.1007/s13042-021-01318-4.

[86] M. Jabi, M. Pedersoli, A. Mitiche, and I. B. Ayed, "Deep Clustering: On the Link Between Discriminative Models and K-Means," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 06, pp. 1887–1896, Jun. 2021, doi: 10.1109/TPAMI.2019.2962683.

[87] H. Li, X. Ye, A. Imakura, and T. Sakurai, "Divide-and-conquer based Large-Scale Spectral Clustering," *ArXiv*, 2021.

[88] J. Wang and J. Jiang, "Unsupervised deep clustering via adaptive GMM modeling and optimization," *Neurocomputing*, vol. 433, pp. 199–211, 2021, doi: 10.1016/j.neucom.2020.12.082.

[89] Z. Kang, X. Lu, J. Liang, K. Bai, and Z. Xu, "Relation-Guided Representation Learning," *Neural Netw.*, vol. 131, pp. 93–102, 2020, doi: 10.1016/j.neunet.2020.07.014.

[90] C. Leiber, L. G. M. Bauer, B. Schelling, C. Böhm, and C. Plant, "Dip-based Deep Embedded Clustering with k-Estimation," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, New York, NY, USA: Association for Computing Machinery, 2021, pp. 903–913. Accessed: Feb. 14, 2022. [Online]. Available: https://doi.org/10.1145/3447548.3467316

[91] F. Li, H. Qiao, and B. Zhang, "Discriminatively boosted image clustering with fully convolutional auto-encoders," *Pattern Recognit.*, vol. 83, pp. 161–173, 2018, doi: 10.1016/j.patcog.2018.05.019.

[92] P. Zhou, Y. Hou, and J. Feng, "Deep Adversarial Subspace Clustering," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 1596–1604. doi: 10.1109/CVPR.2018.00172.

[93] W. Hu, C. Chen, F. Ye, Z. Zheng, and Y. Du, "Learning deep discriminative representations with pseudo supervision for image clustering," *Inf. Sci.*, vol. 568, pp. 199–215, 2021, doi: 10.1016/j.ins.2021.03.066.

[94] L. Wu, Z. Liu, J. Xia, Z. Zang, S. Li, and S. Z. Li, "Generalized Clustering and Multi-Manifold Learning With Geometric Structure Preservation," 2022, pp. 139–147. Accessed: Feb. 15, 2022. [Online]. Available: https://openaccess.thecvf.com/content/WACV2022/html/Wu_Generalized_Clustering_and_Multi-Manifold_Learning_With_Geometric_Structure_Preservation_WACV_2022_paper.html

[95] X. Li, X. Zhao, D. Chu, and Z. Zhou, "An autoencoder-based spectral clustering algorithm," *Soft Comput.*, vol. 24, no. 3, pp. 1661–1671, 2020, doi: 10.1007/s00500-019-03994-5.

[96] E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui, and J. Long, "A Survey of Clustering With Deep Learning: From the Perspective of Network Architecture," *IEEE Access*, vol. 6, pp. 39501–39514, 2018, doi: 10.1109/ACCESS.2018.2855437.

[97] D. B. Graham and N. M. Allinson, "Characterising Virtual Eigensignatures for General Purpose Face Recognition," in *Face Recognition: From Theory to Applications*, H. Wechsler, P. J. Phillips, V. Bruce, F. F. Soulié, and T. S. Huang, Eds. Berlin, Heidelberg: Springer, 1998, pp. 446–456. doi: 10.1007/978-3-642-72201-1_25.

[98] "Face Recognition Grand Challenge (FRGC v.2.0) data collection." [Online]. Available: https://cvrl.nd.edu/projects/data/#face-recognition-grand-challenge-frgc-v20-data-collection

[99] S. A. Nene, S. K. Nayar, and H. Murase, "Columbia Object Image Library (COIL-20)," *Tech. Rep.*, vol. Technical Report CUCS-005-96, 1996.

[100] F. Alimoglu and E. Alpaydin, "Combining multiple representations and classifiers for pen-based handwritten digit recognition," in *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, Aug. 1997, vol. 2, pp. 637–640 vol.2. doi: 10.1109/ICDAR.1997.620583.

[101] M.-F. Balcan, Y. Liang, and P. Gupta, "Robust Hierarchical Clustering," *J. Mach. Learn. Res.*, vol. 15, no. 118, pp. 4011–4051, 2014, Accessed: Feb. 19, 2022. [Online]. Available: http://jmlr.org/papers/v15/balcan14a.html

Jie Yang is currently pursuing the Ph.D. degree in Computational Intelligence and Brain-Computer Interface Lab, Australian AI Institute, FEIT, University of Technology Sydney, Australia. His current research interests include machine learning, clustering, feature selection, and EEG data processing.

Chin-Teng Lin (S'88–M'91–SM'99–F'05) received a Bachelor's of Science from National Chiao-Tung University (NCTU), Taiwan in 1986, and holds Master's and PhD degrees in Electrical Engineering from Purdue University, USA, received in 1989 and 1992, respectively. He is currently a distinguished professor and Co-Director of the Australian Artificial Intelligence Institute within the Faculty of Engineering and Information Technology at the University of Technology Sydney, Australia. He is also an Honorary Chair Professor of Electrical and Computer Engineering at NCTU. For his contributions to biologically inspired information systems, Prof Lin was awarded Fellowship with the IEEE in 2005, and with the International Fuzzy Systems Association (IFSA) in 2012. He received the IEEE Fuzzy Systems Pioneer Award in 2017. He has held notable positions as editor-in-chief of IEEE Transactions on Fuzzy Systems from 2011 to 2016; seats on Board of Governors for the IEEE Circuits and Systems (CAS) Society (2005-2008), IEEE Systems, Man, Cybernetics (SMC) Society (2003-2005), IEEE Computational Intelligence Society (2008-2010); Chair of the IEEE Taipei Section (2009-2010); Distinguished Lecturer with the IEEE CAS Society (2003-2005) and the CIS Society (2015-2017); Chair of the IEEE CIS Distinguished Lecturer Program Committee (2018-2019); Deputy Editor-in-Chief of IEEE Transactions on Circuits and Systems-II (2006-2008); Program Chair of the IEEE International Conference on Systems, Man, and Cybernetics (2005); and General Chair of the 2011 IEEE International Conference on Fuzzy Systems. Prof Lin is the co-author of Neural Fuzzy Systems (Prentice-Hall) and the author of Neural Fuzzy Control Systems with Structure and Parameter Learning (World Scientific). He has published more than 380 journal papers including over 170 IEEE journal papers in the areas of neural networks, fuzzy systems, brain-computer interface, multimedia information processing, cognitive neuro-engineering, and human-machine teaming, that have been cited more than 26,980 times. Currently, his h-index is 75, and his i10-index is 339.