Autonomous clustering by fast find of mass and distance peaks

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

Abstract—Clustering is an essential analytical tool across a wide range of scientific fields, including biology, chemistry, astronomy, and pattern recognition. This paper introduces a novel clustering algorithm as a competitive alternative to existing methods, based on the intuitive principle that a cluster should merge with its nearest neighbor with a higher mass, unless both clusters have relatively large masses and the distance between them is also substantial. By identifying peaks in mass and distance, the algorithm effectively detects and removes incorrect mergers. The proposed method is entirely parameter-free, enabling it to autonomously recognize various cluster types, determine the optimal number of clusters, and identify noise. Extensive experiments on synthetic and real-world data sets demonstrate the algorithm's versatility and consistently strong performance compared to other state-of-the-art methods.

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

1 Introduction and Related Work

Grouping similar objects to derive insights is a fundamental tool in scientific discovery, widely applied across natural and social sciences, including biology, astronomy, psychology, medicine, and chemistry [1]. In data science, this process is known as clustering, a key method for learning from unlabeled data and one of the three broadest categories of machine learning algorithms.

Despite its importance and the abundance of existing algorithms, current clustering methods face several challenges [1]. Numerous efforts have been made to address these issues, with various strategies targeting specific problems. However, no method has yet overcome enough limitations to be considered a universal solution, as optimal clustering remains an NP-hard problem. As a result, researchers often test and fine-tune multiple algorithms to find the most suitable one [2].

Standard hierarchical clustering, for example, has high computational costs with a time complexity of $O(n^3)$, making it unsuitable for large-scale data sets. Additionally, manually determining when to stop the clustering process, such as choosing the number of clusters, remains a challenge. Some newer algorithms accelerate the process using fast approximate nearest neighbor methods [3], [4], while others prune cluster trees to estimate the correct number of clusters [5], [6].

Partitioning methods like K-means [7] require the number of clusters to be known or estimated in advance, struggle to detect non-convex clusters, and are sensitive to noise, outliers, and initialization. Improvements, such as X-means [8], can estimate the number of clusters automatically, and kernel K-means variants address the non-convexity problem [9], while other methods aim to improve initialization [10], [11].

Density-based clustering traditionally required manually set thresholds, such as cutoff distances or minimum points for high-density clusters. Modern approaches reduce reliance on these thresholds [12], [13], [14], [15]. For instance, Liang et al. [12] introduced 3DC clustering to automatically determine the number of clusters, while Du et al. [15] incorporated k-nearest neighbors (KNN) and principal component analysis into density peak clustering to improve accuracy on high-dimensional data sets.

Model-based clustering often relies on prior knowledge of cluster parameters, which can be difficult to obtain. Recent solutions aim to mitigate this limitation [16], [17], [18]. For example, Scrucca et al. [16] improved model-based clustering through data transformations, enhancing model fit and clustering accuracy, while O'Hagan et al. [17] proposed a Bayesian method for generating high-quality initial parameters for expectation–maximization algorithms.

Finally, grid-based clustering depends on user-provided parameters like interval values and density thresholds, and most algorithms struggle to scale with high-dimensional data sets. Various variants have been proposed to address these issues [19], [20], [21]. For example, Chen et al. [19] developed a new grid-based clustering algorithm for hybrid data streams that automatically determines the number, center, and radius of clusters.

From the overview of mainstream clustering algorithms mentioned above, an effective clustering algorithm should be capable of identifying clusters of varying shapes, sizes, and densities, be parameter-free, have reasonable computational efficiency, be robust to noise and outliers, and not require manually specified stopping conditions.

To meet these requirements, we propose Torque Clustering (TC), a novel algorithm inspired by the natural process of galaxy mergers. TC operates on the principle that a cluster should merge with its nearest neighbor that has a higher mass, unless both clusters are large and the distance

- *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).*

between them is substantial. This process mirrors the gravitational interactions observed in galaxy mergers, where larger galaxies merge with smaller ones, creating a hierarchical structure similar to galaxy merger trees documented by astronomers [22]. TC models this minor merger process, continuously merging smaller adjacent clusters into larger ones and generating a hierarchical tree that reflects the natural structure of the data.

After constructing this hierarchical tree, TC prunes it to estimate the correct number of clusters. Unlike existing methods that rely on probability density [6], TC uses mass and distance peaks to determine cluster partitions, a concept inspired by the vast distances between large galaxies in the universe [23]. The product of cluster masses ($m_1 m_2$) and the square of the distance between clusters (r^2) are used to evaluate each merger, and mergers with relatively large values are removed, resulting in the final cluster structure.

We evaluated TC on 21 data sets across six domains—image recognition, biology, medicine, physics, natural language processing, and astronomy—comparing its performance against over a dozen state-of-the-art algorithms. In terms of accuracy, TC ranked first on 15 of the 20 data sets, outperforming the best competing algorithm by a factor of 3 in average ranking on real-world data sets. Furthermore, we compared TC with nine advanced parameter-free or automatic clustering algorithms in terms of their ability to automatically determine the number of clusters, where TC achieved the lowest average error. Additionally, we performed a comprehensive evaluation on 56 data sets with noise, outliers, overlaps, imbalance, and high dimensionality, where TC demonstrated a notable performance advantage across these challenging scenarios. Finally, in comparisons with recent deep clustering algorithms on several high-dimensional image data sets, TC—despite not utilizing deep representations—achieved results comparable to those of deep clustering methods, underscoring TC's robustness in high-dimensional data contexts.

This paper is organized as follows: In Section 2, we introduce the TC algorithm, detailing its specific procedures and providing an in-depth analysis. Section 3 offers a comprehensive evaluation of TC, comparing it with over a dozen state-of-the-art clustering algorithms on both synthetic and real-world data sets. Section 4 discusses the advantages of TC over other hierarchical and density peak clustering algorithms, along with its potential limitations. Finally, Section 5 and Section 6 conclude the paper by summarizing our findings and proposing potential directions for future research.

2 THE ALGORITHM

In this section, we provide a detailed explanation of the specific procedures of TC, including: defining clusters and forming connections between them, defining two properties of each connection to construct the decision graph, defining the torque of each connection and sorting the connections in descending order, defining the torque gap and identifying the largest gap to detect abnormal connections, and defining halo connections to identify noise. Additionally, we conduct a thorough complexity analysis of TC and present visual comparisons with related methods. 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 $X = \{x_1, x_2, \ldots, x_n\}$, where $x_i \in \mathbb{R}^D$. The first step is to determine the initial mass of the data set, which corresponds to the number of points. Initially, each data point x_i is treated as a separate cluster ζ_i, forming the initial cluster set $\Gamma = \{\zeta_1, \zeta_2, \ldots, \zeta_n\}$, representing the first layer of the hierarchical tree. The mass of each cluster is recorded in the set $\Theta = \{\theta_1, \theta_2, \ldots, \theta_n\}$, where all clusters are assigned an initial mass of 1 (i.e., $\theta_i = 1$ for every i). The following rule is then applied to establish connections between clusters:

$$\zeta_i \rightarrow \zeta_i^N, \quad if\ \theta_i \leq \theta_i^N \tag{1}$$

where ζ_i^N denotes the 1-nearest cluster to ζ_i, and θ_i^N represents the number of points within ζ_i^N. The symbol "→" indicates the connection (i.e., merger) C_i between ζ_i and ζ_i^N. Treating each cluster as a vertex, a connected graph G can be constructed.

A new set of clusters, Γ', is then formed as follows:

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

where $\Phi(\,)$ identifies the points within each connected component as a new cluster. Accordingly, the mass of each new cluster remains the count of points it contains.

By applying Eq. (1) to Γ', new connections C_i are formed, altering the connected graph G. Then, using Eq. (2) on G generates the next cluster set Γ', and the process continues until a single, all-encompassing cluster is formed at the top of the hierarchical tree. Unlike classic agglomerative hierarchical clustering, this constrained merging method avoids the formation of elongated clusters and can be performed in parallel as long as neighboring clusters meet the criteria of Eq. (1). This iterative merging achieves three goals: it creates a hierarchical map of clusters at varying granularities, provides the algorithm with a structure to select the most appropriate clustering scheme, and allows analysts to manually adjust the clustering granularity if desired. To determine the "most appropriate" clustering scheme, the algorithm identifies and removes "abnormal" connections—those that connect clusters with both relatively large masses and long distances between them—revealing a more reasonable partitioning. The following steps describe the mechanisms for detecting and removing these abnormal connections.

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

Abnormal connections can be identified by examining two intuitive properties of the connection C_i: one is the product of the mass values of the two clusters it connects, defined as

$$M_i = \theta_i \times \theta_i^N \tag{3}$$

and the other is the square of the distance between these two clusters,

$$D_i = d^2(\zeta_i, \zeta_i^N) \tag{4}$$

Plotting all connections on a two-dimensional decision graph with these properties reveals that the mass and distance of abnormal connections are significantly larger than and more distant from those of typical connections. Figure 1 illustrates the core concept of the proposed TC algorithm.

Many studies have explored how to define distance between clusters [24]. Here, we define it simply as the minimum distance from any point in one cluster to any point in the other, i.e.,$d(\zeta_i, \zeta_i^N) = \min_{x_a \in \zeta_i, x_b \in \zeta_i^N} d(x_a, x_b)$. For large-scale data sets, a fast approximate nearest neighbor method, such as k-d tree or locality-sensitive hashing, may be more appropriate for identifying the nearest clusters, as these methods avoid the need to compute pairwise distances between all clusters [4].

A detailed explanation of how TC operates is provided through a step-by-step example in Figure 2 and Table 1.

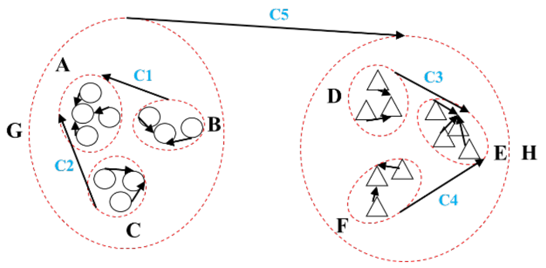
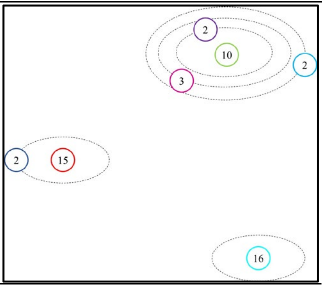


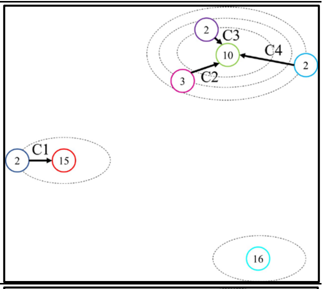
Figure 1. The core idea of the proposed algorithm. The red dotted lines delineate clusters A-H in a two-dimensional data distribution, derived from Eqs. (1) and (2). The black lines C_1 to C_5 represent the connections between each cluster and its nearest neighbor, with lengths L_i, where L_5 is the longest. Each connection links two clusters containing multiple points. For instance, clusters A and E each contain four points; B, C, D, and F contain three; and G and H contain 10 points. The goal is to identify "abnormal" connections, defined as those with both a relatively large distance (i.e., D_i in Eq. (4)) and a relatively large number of points (i.e., M_i in Eq. (3)). Connection C_5, which links two clusters with 10 points each, carries the greatest mass and is the longest (L_5). What is key here is relativity: C_5 is notably longer than the other connections. Removing C_5 and recalculating the connected components using Eq. (2) results in a more intuitive and reasonable set of clusters, consistent with both human intuition and the natural laws governing galaxy interactions.

Figure 2. A stage-by-stage example of how TC works. Consider a data set where, initially, each point 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 points within them (the value in the circles).

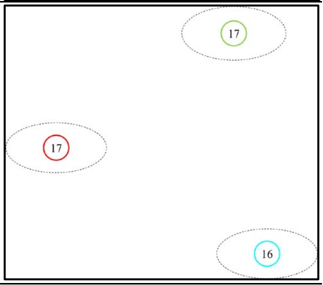
A
Applying Eq. (2) to the connected graph of the initial clusters reveals seven new larger clusters.



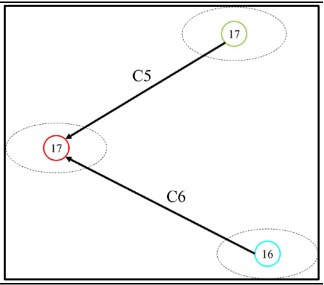
B
Connections C_1-C_4 can then be added according to the adjacency relationship given by Eq. (1). Now, the two properties M_i and D_i of C_1-C_4 can be calculated according to Eqs. (3)- (4), as shown in Table 1A.



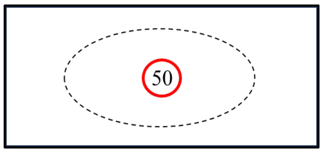
C
Again, applying Eq. (2) to the connected graph of Fig. 2B reveals three new larger clusters. The mass of each new cluster is equal to the sum of the masses of the sub-clusters it contains.



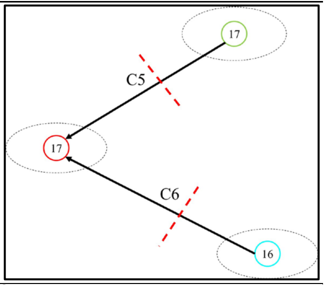
D
Connections C_5-C_6 can then be added according to the adjacency relationship given by Eq. (1). Now, the two properties M_i and D_i of C_5-C_6 can be calculated Eqs. (3)- (4), as shown in Table 1B.



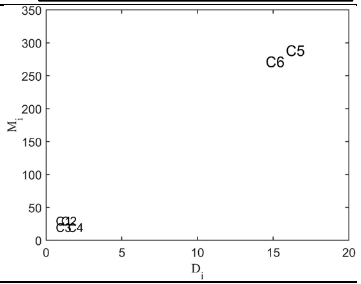
E
Again, applying Eq. (2) to the connected graph of Fig. 2D, we now have one big cluster, and the merging process is complete. Steps A-E show that this process has established a hierarchical tree of clustering partitions at different granularities, as illustrated in Fig. 2H.



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.

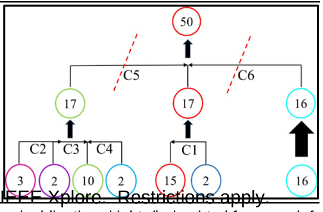


TABLE 1A. Properties of the clusters and connections in Figs. 2A and 2B.

Cluster	Mass	Nearest Cluster	Mass	Connect?	M_i	D_i	Connection Number
🔴	15	🔵	2	×	–	–	–
🔵	2	🔴	15	✓	30	0.64	C1
🟢	10	🟣	2	×	–	–	–
🩷	3	🟢	10	✓	30	1.00	C2
🟣	2	🟢	10	✓	20	0.64	C3
🔵	2	🟢	10	✓	20	1.44	C4
🩵	16	🔴	15	×	–	–	–

TABLE 1B. Properties of the clusters and connections in Figs. 2C and 2D.

Cluster	Mass	Nearest Cluster	Mass	Connect?	M_i	D_i	Connection Number
🔴	17	🩵	16	×	–	–	–
🟢	17	🔴	17	✓	289	15.83	C5
🩵	16	🔴	17	✓	272	14.50	C6

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

The decision graph generated by TC provides an efficient visualization tool to identify and remove abnormal connections. However, manually inspecting the 2D plot can be error-prone and time-consuming. To address this, we propose an automatic method for identifying abnormal connections based on a metric that measures the gaps between connections, termed the Torque Gap (*TGap*) due to its mathematical similarity to torque. The *TGap* calculation begins by determining the torque, τ_i, for each connection:

$$\tau_i = M_i \times D_i \tag{5}$$

where M_i is the product of the masses of the two clusters connected, and D_i is the squared distance between them. Naturally, if two clusters connected by a connection have relatively large masses and a significant distance between them, the connection's torque, τ_i, will be substantial.

We then sort all connections in descending order by their torque values to create the Torque Sorted Connections List (TSCL). In the TSCL, each connection and its torque are represented as $\grave{C}_i$ and $\grave{\tau}_i$, respectively.

Our core concept is that abnormal connections should appear at the top of the TSCL due to their high torque values. However, determining precisely which connections are "abnormal" requires calculating the *TGap* within the TSCL.

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

To compute the *TGap* between consecutive connections in the TSCL, we use the formula:

$$TGap_i = \omega_i \times \frac{\grave{\tau}_i}{\grave{\tau}_{i+1}},\ \grave{\tau}_{i+1} \neq 0 \tag{6}$$

where ω_i is a weight indicating the proportion of connections among the top i TSCL connections that possess relatively large values of M_i, D_i, and τ_i.

Defining ω_i involves the following steps: Eq. (1) reveals numerous connections, C_i, throughout the clustering process, each with two properties, M_i and D_i. We define the set of connections with relatively high values of M_i, D_i, and τ_i (denoted as $Large\_C$) as:

$$Large\_C = \{C^* | \tau_{C^*} \geq mean\_\tau, M_{C^*} \geq mean\_M, D_{C^*} \geq mean\_D\} \tag{7}$$

where $mean\_\tau$, $mean\_M$, and $mean\_D$ represent the average values of all τ_i, M_i, and D_i, respectively.

The set of the top i TSCL connections, Top_C_i, is defined as:

$$Top\_C_i = \{\grave{C}_1, \grave{C}_2, \ldots, \grave{C}_i\} \tag{8}$$

Using $Large\_C$ and Top_C_i, we define ω_i as:

$$\omega_i = \frac{|Large\_C \cap Top\_C_i|}{|Large\_C|} \tag{9}$$

The largest $TGap_i$ is denoted as $TGap_L$, and the L connections at the top of the TSCL (i.e., $\{\grave{C}_1, \grave{C}_2, \ldots, \grave{C}_L\}$) are regarded as abnormal connections to be removed. Figure 2I illustrates this procedure.

The $TGap$ in Eq. (6) considers two key factors for identifying abnormal connections: $\frac{\grave{\tau}_i}{\grave{\tau}_{i+1}}$, which is the natural torque gap between consecutive connections in the TSCL, and ω_i, representing the clustering resolution. A larger torque gap indicates a more appropriate point for cutting connections. The purpose of calculating $\frac{\grave{\tau}_i}{\grave{\tau}_{i+1}}$ is to detect the distinction between connections with higher and lower torque values. The clustering resolution factor ω_i accounts for uneven cluster distribution or imbalance in the data set. For instance, consider a data set with three relatively balanced ground-truth clusters, $\{A\}, \{B\}, \{C\}$. After applying TC, we identify connections C_{AB} between A and B, and C_{BC} between B and C. Suppose the distance between A and B is significantly larger than between B and C, and both are much larger than the intra-cluster distances. If we only rely on $\frac{\grave{\tau}_i}{\grave{\tau}_{i+1}}$, we might remove only C_{AB}, resulting in an incorrect partition: $\{A\}, \{B, C\}$. However, we can consider both $\frac{\grave{\tau}_i}{\grave{\tau}_{i+1}}$ and ω_i at the same time. Since the M_{BC}, D_{BC}, and τ_{BC} of C_{BC}, are also large relative to other connections except C_{AB} (i.e., $C_{BC} \in Large\_C$), then ω_2 is more likely to be greater than ω_1, which may ensure that $TGap_2$ is the largest. Therefore, 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 a critical step. In our algorithm, we define an additional type of connection specifically for identifying noise, referred to as "halo connections" (denoted as Halo_C). Halo connections are distinguished by a relatively large D_i and a relatively small M_i. The formula for identifying halo connections is as follows:

$$Halo\_C = \left\{C^{\aleph} | M_{C^{\aleph}} \leq mean\_M, D_{C^{\aleph}} \geq mean\_D, \frac{D_{C^{\aleph}}}{M_{C^{\aleph}}} \geq mean\_\frac{D}{M}\right\} \tag{10}$$

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

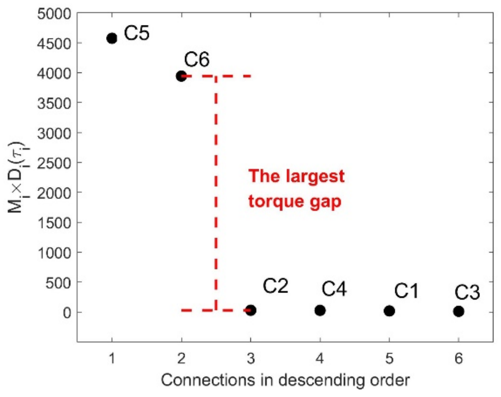


Fig. 2I. Identifying abnormal connections using the largest torque gap. After calculating the torque of each connection generated from Figs.2A-E, all connections are sorted in the order of decreasing torque to obtain the TSCL, i.e., $C_5, C_6, C_2, C_4, C_1, C_3$. Furthermore, Eqs. (6)-(9) are applied 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.

In Section 2.4, after removing the L abnormal connections, $L+1$ clusters are formed. At this stage, halo connections are further removed, allowing for the identification of small sub-clusters within the $L+1$ clusters, which are then considered part of the cluster halo (suitable for designation as noise). Figure 3 provides an example illustrating the effectiveness of abnormal and halo connections. The pseudocode for TC is presented in Algorithm 1.

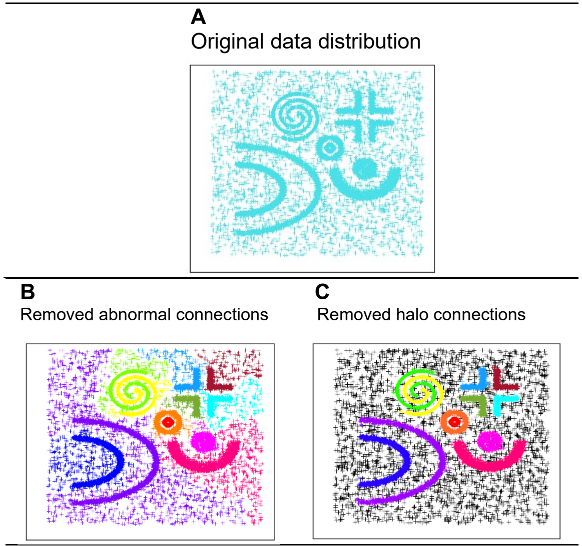


Figure 3. TC on the synthetic data set with 30% uniform noise. (A) is the original data distribution, including convex and non-convex clusters with 30% uniform noise; (B) illustrates the results of TC after removing the abnormal connections; (C) illustrates the results of TC after removing the halo connections, 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 cost of steps 6, 8 and 9 is

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 point as a cluster, i.e., $l = n$).
5: while cluster set $\Gamma$ have more than two clusters do
6 : Computing the mass $\theta_i$ of each cluster in $\Gamma$, where $\Theta = \{\theta_i\}_{i=1}^{l}$ .
7: Searching the nearest cluster of $\zeta_i$ 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 $\Gamma$ by Eq. (2).
11: end
12: Computing the torque $\tau_i$ 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.
```

$O(mn)$, 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 point as its own cluster, so the distances between neighboring data points can be regarded as the distances between neighboring clusters in the first loop without extra computing. Therefore, the total cost of step 7 is $O((m-1)l^2)$. For step 10, since m loops generate a total of n-1 connections, its time cost is $O(mn+n-1) = O((m+1)n-1)$. Steps 12, 14, 15 and 17 each require $O(n)$, and step 13 requires $O(nlogn)$. The time cost 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, the time complexity of TC is approximately $O(n^2)$.

However, when using a fast approximate nearest neighbor method, we don't need to compute the distance matrix S, so the time cost of step 1 becomes 0 and $O(ml * logl)$ for step 7, the time complexity of TC here is approximately $O(nlogn)$.

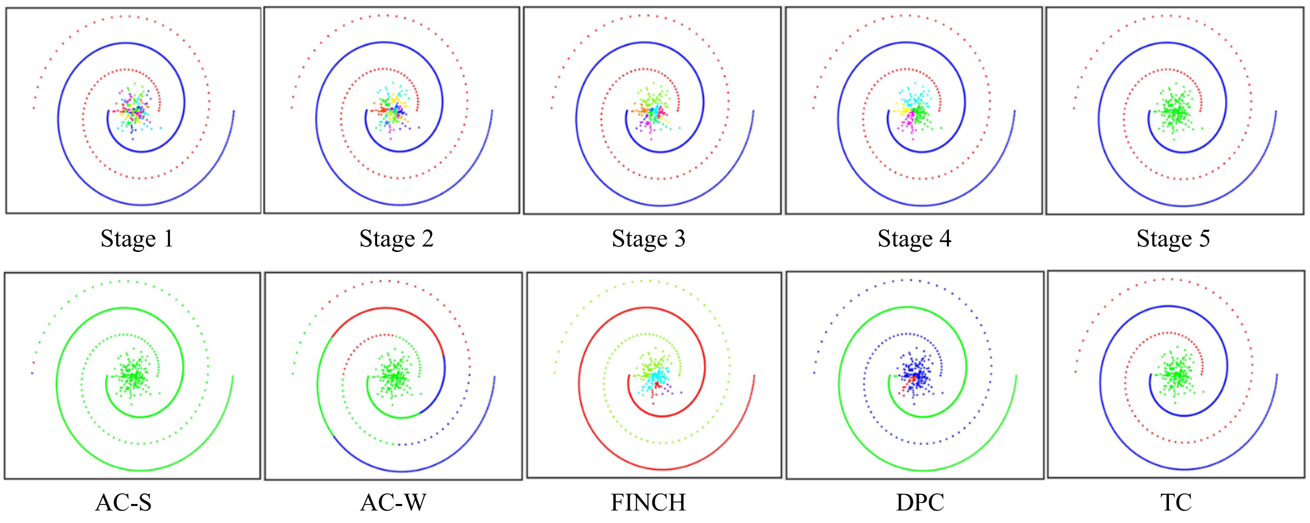


Figure 4. Visualization of the stage-by-stage results of TC (top) and the final partitions of related methods (bottom) on a synthetic data set.

- **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, the base space requirement is $O(n)$. 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 $O(n^2)$.

2.7 Algorithm Analysis

For clarity, Fig. 4 visualizes the stage-by-stage results of TC and the final outcomes of several related methods on a synthetic data set. TC follows the merging rule outlined in Eq. (1), gradually combining clusters and automatically determining the correct number of clusters. In stage 1, the red and blue clusters are correctly formed, matching the ground truth. In stages 2-5, due to the constraint in Eq. (1), the red and blue clusters do not merge with their nearest neighbors, waiting instead for other sub-clusters to merge. This approach effectively prevents potential incorrect mergers, which is common in traditional agglomerative methods.

In contrast, agglomerative clustering with single-linkage (AC-S) is sensitive to outliers, resulting in inaccurate cluster formation, while agglomerative clustering with Ward linkage (AC-W) struggles to detect clusters with complex shapes. FINCH [4], a nearest-neighbor-based agglomerative method without constraints, also produces some incorrect mergers. Density peak clustering (DPC) [25] has difficulties handling data sets with varying densities, leading to further errors. Additionally, all other methods require manual input to determine the number of clusters (or granularity levels), whereas TC is fully automatic.

3 Experiments and Results

In this section, we evaluate the proposed TC algorithm, highlighting its effectiveness and robustness across diverse data sets. We outline the experimental setup, compare TC with 16 advanced clustering algorithms across 21 synthetic and real-world data sets, and extend the assessment by including newer algorithms and additional data sets for a more comprehensive evaluation. Finally, we present an analysis of the results to offer further insights into TC's performance.

3.1 Experimental Setup

To evaluate TC's performance, we tested it on a range of synthetic and real-world data sets, comparing its results with those of 16 widely used or recent clustering algorithms across various types, including: K-means++ (K-M++) [10], spectral clustering (SC) [26], [27], and hierarchical agglomerative clustering with single-linkage (AC-S), complete-linkage (AC-C), average-linkage (AC-A), ward-linkage (AC-W), and centroid-linkage (AC-CR) [28]; density peak clustering (DPC) [25] and its three recent variants—dynamic graph-based label propagation for density peak clustering (DPCLP) [29], shared-nearest-neighbor-based density peak clustering (SNNDPC) [30], and automatic density peak clustering (DPA) [31]; as well as efficient parameter-free clustering using first neighbor relations (FINCH) [4], DBSCAN (DB) [32], affinity propagation (AP) [33], border-peeling clustering (BP) [34], and robust continuous clustering (RCC) [1]. Notably, DPA, FINCH, DB, AP, BP, and RCC, like TC, are automatic clustering algorithms. For each experiment, we optimized free parameters in the baseline methods across a broad range of configurations to maximize their performance, a significant advantage for these methods. Implementation details for each baseline algorithm are provided in the supplementary materials, while TC's reported performance is based on a single run, with 1-nearest clusters implemented using a distance matrix. Each experiment was assessed using three widely used external indices: normalized mutual information (NMI) [35], accuracy (ACC), and adjusted mutual information (AMI) [36]. Finally, following common practice, we used average rank across data sets for each index, with a lower rank indicating better overall performance.

TABLE 2. Statistics of the 21 data sets.

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
YTF	10056	9075	40	~13
MNIST	70000	784	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
Reuters	8067	2000	30	~206

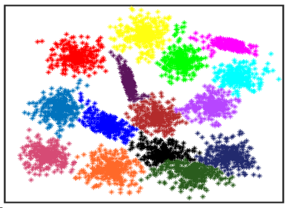
3.2 Comparison with 16 Diverse Algorithms on Nine Synthetic Data sets

In this part, we evaluated TC on nine synthetic data sets, which are designed to reflect seven common clustering challenges: overlap, FLAME, spectral-path, unbalanced, noisy, heterogeneous geometric, and multi-objective. Table 2 provides descriptive statistics for these data sets.

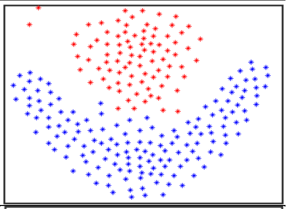
The visualization results and data set descriptions are shown in Figs. 5A-5I, demonstrating TC's effectiveness in handling each challenge. Notably, TC automatically determined the exact number of clusters for all nine data sets. For visual comparison, K-means was also applied, as shown in the supplement, where K-means failed on eight out of nine data sets. Full quantitative comparisons with 16 clustering algorithms of various types are presented in Tables 3A, 3C, and 3E, showing that TC achieved the highest performance on seven data sets for the three evaluation indices, with an overall top ranking in average performance.

Figure 5. TC results on nine synthetic data sets reflecting different clustering challenges.

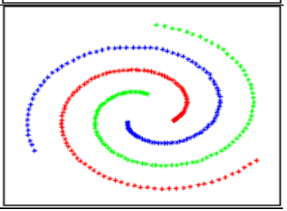
A.[37] **Highly overlapping:** TC successfully identified 15 clusters in this data set, despite substantial overlaps.



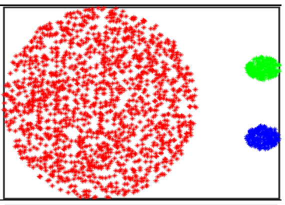
B.[38] **FLAME:** TC detected the two clusters in this data set, which was originally designed to test fuzzy clustering via local membership approximation (FLAME).



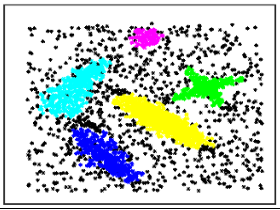
C.[39] **Spectral-path:** This data set is often used to illustrate path-based spectral clustering performance. TC was able to identify the three clusters without requiring a path-based connectivity graph.



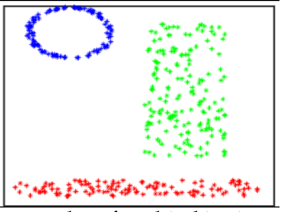
D.[40] **Unbalanced:** Despite severe class imbalances, TC identified the clusters effectively, as shown by the significantly disproportionate clusters on the right.



E.[25] **Noisy:** Originally intended to showcase how density-based algorithms handle noise, TC detected the five clusters in this noisy data with reasonable accuracy.

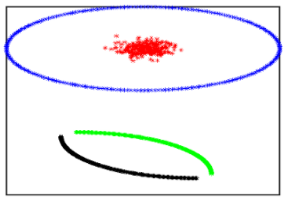


F.[41] **Heterogeneous geometric:** TC identified three clusters intuitively, without needing point symmetry distance calculations required by some methods.

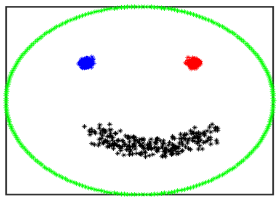


Multi-objective: Figs. 5G-5I illustrate examples of multi-objective clustering, where typically, multiple algorithms or ensemble learning are employed to reveal distinct cluster structures. TC was able to capture these different structures without specialized adaptations.

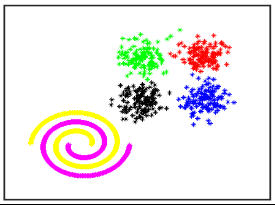
G. [42]



H. [43]



I. [44]



3.3 Comparison with 16 Diverse Algorithms on 12 Real-World Data sets

In this part, we evaluated TC on 12 additional real-world clustering tasks across six domains: image recognition, biology, medicine, physics, natural language processing (NLP), and astronomy. Comprehensive descriptive statistics for these data sets are presented in Table 2. Below, we detail each task and TC's performance on them, with Table 3 providing a full quantitative comparison against 16 other clustering algorithms.

The image recognition tasks included handwritten digit recognition, face recognition, object recognition, and PIE (Pose, Illumination, Expression) recognition as four distinct tasks. For digit recognition, we used the MNIST data set [45], preprocessed following the method outlined in prior research [46]. TC achieved top scores across all indices, classifying digits with an ACC of 97.35%.

The face recognition task utilized the YouTube Faces Database (YTF) with images of various individuals [1]. TC

TABLE 3A. Comparison with 16 diverse algorithms on synthetic data sets, measured by AMI. The Noisy synthetic data set lacks ground-truth labels, so we excluded it from the comparison to maintain objectivity.

Data sets/Methods	K-M++	SC	AC-A	AC-W	AC-S	AC-C	AC-CR	DPC	DPCLP	SNNDPC	DPA	FINCH	DB	AP	BP	RCC	TC
Highly overlapping	.9295	.0042	.9592	.9344	.0346	.8747	.9288	**.9744**	.8404	.9569	.9702	.7634	.2830	.5614	.7753	.8150	.9562
FLAME	.4117	.0088	.4687	.3269	.0088	.0571	.0088	.4031	.7781	.8165	.3463	.2279	.7410	.2017	.8844	.9267	**1.0000**
Spectral-path	-.0053	1.0000	-.0029	.0009	1.0000	.0046	.0060	1.0000	.2285	1.0000	.3154	.2445	1.0000	.2855	.1785	.3279	**1.0000**
Unbalanced	.3983	1.0000	.6108	.6109	1.0000	.3391	.6351	1.0000	.6228	1.0000	.6526	.1247	1.0000	.1388	.2617	.1563	**1.0000**
Heterogeneous geometric	.7321	1.0000	1.0000	1.0000	1.0000	.4324	1.0000	.7433	1.0000	1.0000	1.0000	.2916	1.0000	.3299	.6387	.6554	**1.0000**
Multi-objective 1	.5799	1.0000	.5710	.5841	.7492	.5821	.5821	.7492	.9670	1.0000	.7673	.4821	.9953	.3829	.7860	1.0000	**1.0000**
Multi-objective 2	.6025	1.0000	.6547	.6033	1.0000	.6579	.6307	.6501	1.0000	1.0000	.7495	.4481	1.0000	.4296	.6720	.8998	**1.0000**
Multi-objective 3	.5573	.7343	.7004	.7213	.7055	.6972	.7188	**.9950**	.6314	.7134	.6005	.7947	.9557	.3426	.8882	.4710	.9925
Rank	*12.0*	*5.4*	*9.4*	*9.4*	*6.8*	*12.0*	*9.8*	*5.6*	*7.0*	*2.8*	*7.2*	*13.0*	*4.1*	*15.0*	*9.4*	*9.0*	***1.6***

TABLE 3B. Comparison with 16 diverse algorithms on real-world data sets, measured by AMI. "NA" means not applicable.

Data sets/Methods	K-M++	SC	AC-A	AC-W	AC-S	AC-C	AC-CR	DPC	DPCLP	SNNDPC	DPA	FINCH	DB	AP	BP	RCC	TC
YTF	.7145	.4622	.4463	.7612	.3573	.4812	.4016	.7603	.3068	.5860	.6306	.7531	.7361	.5373	.7342	.8025	**.8384**
MNIST	.8839	.8386	.8889	NA	.8775	.8888	NA	.9292	NA	NA	.3806	.4576	.8390	.3446	.5171	.8021	**.9295**
COIL-100	.7614	.8007	.5877	.7891	.4685	.6453	.4763	.7918	.0000	.5829	.7914	.6643	.5176	.6080	.3162	.9237	**.9439**
Shuttle	.2321	.5899	.0068	.2198	.0036	.0516	.0068	.5769	NA	NA	.3700	.0368	.5113	.3270	.3273	.4858	**.6389**
RNA-seq	.8707	.9944	.0061	.9851	.0001	.6459	.0001	.7583	.0000	.9888	.9545	.7706	.4803	.4362	.8530	.9944	**.9944**
Haberman	.0147	.0028	-.0033	.0116	.0043	-.0033	-.0006	.0084	.0030	-.0029	.0181	.0197	.0280	.0198	.0115	.0305	**.0633**
Zoo	.7468	.6558	.7733	.7073	.2990	.6991	.7733	.6720	.1892	.7849	.2603	.8190	.6821	.5764	.0041	**.8622**	.8546
Atom	.2414	.0013	.1781	.1781	1.0000	.1621	.0246	.2305	1.0000	1.0000	.6779	.3034	1.0000	.1839	.4053	.2858	**1.0000**
Soybean	.8908	1.0000	.7322	1.0000	1.0000	1.0000	.7322	.8249	.7282	1.0000	.3647	.7047	.7322	.7494	.0000	1.0000	**1.0000**
Cell-track	.5014	.5360	.4696	.4696	.5360	.4696	.4696	.0881	.0399	-.0171	.5360	.5360	**.6133**	.2758	.0273	.1098	.5360
CMU-PIE	.4238	.7381	.3586	.4889	.9797	.3628	.2213	.9183	.1340	.2622	.8608	.5953	.6280	.6146	.0536	.6868	**1.0000**
Reuters	.3373	.4339	.3747	.3618	.0383	.3489	.2726	.3343	.0826	.1019	.3338	.4547	.3410	.2040	.0472	**.5484**	.4770
Rank	*7.9*	*6.8*	*10.0*	*7.7*	*9.6*	*9.6*	*13.0*	*7.6*	*14.0*	*10.0*	*8.2*	*7.5*	*6.9*	*11.0*	*12.0*	*4.3*	***1.2***

TABLE 3C. Comparison with 16 diverse algorithms on synthetic data sets, measured by NMI.

Data sets/Methods	K-M++	SC	AC-A	AC-W	AC-S	AC-C	AC-CR	DPC	DPCLP	SNNDPC	DPA	FINCH	DB	AP	BP	RCC	TC
Highly overlapping	.9362	.0128	.9596	.9354	.0346	.8867	.9455	**.9747**	.8497	.9572	.9705	.8665	.2830	.7542	.8460	.8317	.9568
FLAME	.4252	.0479	.4832	.3297	.0479	.0770	.0479	.4132	.7937	.8288	.5805	.4896	.8374	.4486	.9083	.9269	**1.0000**
Spectral-path	.0006	1.0000	.0031	.0068	1.0000	.0106	.0119	1.0000	.3037	1.0000	.3903	.5359	1.0000	.5549	.2056	.5940	**1.0000**
Unbalanced	.4399	1.0000	.6108	.6109	1.0000	.4406	.6351	1.0000	.6228	1.0000	.6526	.3720	1.0000	.3842	.5134	.4101	**1.0000**
Heterogeneous geometric	.7371	1.0000	1.0000	1.0000	1.0000	.4611	1.0000	.7445	1.0000	1.0000	1.0000	.5583	1.0000	.5856	.8017	.8127	**1.0000**
Multi-objective 1	.6186	1.0000	.6895	.5981	.8633	.7008	.6636	.8035	.9673	1.0000	.8766	.6995	.9977	.6267	.8750	1.0000	**1.0000**
Multi-objective 2	.6169	1.0000	.6894	.6130	1.0000	.6920	.6775	.6663	1.0000	1.0000	.8660	.6846	1.0000	.6632	.7999	.9489	**1.0000**
Multi-objective 3	.5850	.7881	.7020	.7229	.8341	.7052	.7204	**.9950**	.6868	.7304	.7030	.7947	.9709	.6015	.9092	.6943	.9925
Rank	*14.0*	*5.5*	*9.6*	*11.0*	*6.1*	*12.0*	*10.0*	*6.6*	*7.5*	*2.8*	*6.8*	*11.0*	*3.9*	*14.0*	*9.0*	*8.8*	***1.6***

TABLE 3D. Comparison with 16 diverse algorithms on real-world data sets, measured by NMI.

Data sets/Methods	K-M++	SC	AC-A	AC-W	AC-S	AC-C	AC-CR	DPC	DPCLP	SNNDPC	DPA	FINCH	DB	AP	BP	RCC	TC
YTF	.7368	.4622	.4463	.7811	.3573	.5828	.4016	.7947	.3068	.6540	.6794	.7757	.7528	.7695	.8060	**.8978**	.8604
MNIST	.8839	.8386	.8889	NA	.8775	.8888	NA	.9292	NA	NA	.6020	.6560	.8390	.5741	.5171	.8021	**.9295**
COIL-100	.8137	.8564	.7256	.8353	.6990	.7428	.6577	.8657	.0419	.6919	.8504	.7897	.7223	.8393	.5510	.9628	**.9720**
Shuttle	.2321	.5860	.0277	.2671	.0385	.0516	.0285	.5769	NA	NA	.3700	.0368	.5113	.3270	.3273	.4858	**.6389**
RNA-seq	.8918	.9948	.0464	.9860	.0318	.7252	.0320	.8302	.0000	.9895	.9745	.8785	.5620	.6699	.8960	.9948	**.9948**
Haberman	.0200	.0057	.0006	.0204	.0387	.0006	.0083	.0114	.0118	.0000	.0344	.0295	.0448	.0666	.0249	.0720	**.0847**
Zoo	.7946	.7396	.8331	.7615	.5391	.7389	.8331	.7218	.3473	.8481	.4362	.8654	.7911	.7717	.0770	**.8979**	.8913
Atom	.2833	.0158	.2257	.2257	1.0000	.2107	.0620	.2735	1.0000	1.0000	.8236	.5560	1.0000	.4382	.6382	.5503	**1.0000**
Soybean	.9180	1.0000	.8264	1.0000	1.0000	1.0000	.8264	.8635	.7554	1.0000	.6179	.8489	.8264	.8817	.0000	1.0000	**1.0000**
Cell-track	.5138	.5466	.4835	.4835	.5466	.4835	.4835	.1577	.0631	.0025	.5466	.5466	**.6211**	.4115	.0970	.4387	.5466
CMU-PIE	.5406	.8123	.5060	.5982	.9897	.4920	.3859	.9532	.3857	.3930	.9152	.7977	.7589	.8387	.2099	.8208	**1.0000**
Reuters	.4492	.4339	.3747	.4834	.0383	.3489	.2726	.3343	.0826	.1019	.3338	.4547	.3410	.4717	.0472	**.5484**	.4770
Rank	*8.3*	*7.2*	*11.0*	*7.8*	*8.8*	*10.0*	*13.0*	*8.1*	*14.0*	*11.0*	*8.2*	*7.6*	*7.4*	*8.2*	*12.0*	*4.2*	***1.4***

TABLE 3E. Comparison with 16 diverse algorithms on synthetic data sets, measured by ACC.

Data sets/Methods	K-M++	SC	AC-A	AC-W	AC-S	AC-C	AC-CR	DPC	DPCLP	SNNDPC	DPA	FINCH	DB	AP	BP	RCC	TC
Highly overlapping	.9090	.0773	.9754	.9570	.0738	.8514	.9138	**.9856**	.8548	.9740	.9842	.7468	.1334	.3124	.7318	.6692	.9714
FLAME	.8401	.6458	.8333	.7208	.6458	.5167	.6458	.7875	.9625	.9708	.5375	.2292	.9375	.1458	.9833	.9917	**1.0000**
Spectral-path	.3458	1.0000	.3590	.3750	1.0000	.3878	.4038	1.0000	.5769	1.0000	.4744	.1571	1.0000	.1635	.5032	.2692	**1.0000**
Unbalanced	.4761	1.0000	.5435	.5440	1.0000	.5320	.6795	1.0000	.6325	1.0000	.5160	.1015	1.0000	.0670	.3510	.2370	**1.0000**
Heterogeneous geometric	.8833	1.0000	1.0000	1.0000	1.0000	.6075	1.0000	.8500	1.0000	1.0000	1.0000	.1350	1.0000	.1700	.7700	.7200	**1.0000**
Multi-objective 1	.5926	1.0000	.5890	.6580	.7490	.6180	.5760	.6500	.9890	1.0000	.7450	.4280	.9990	.1490	.8620	1.0000	**1.0000**
Multi-objective 2	.7050	1.0000	.8070	.7140	1.0000	.8090	.7830	.7320	1.0000	1.0000	.7500	.5490	1.0000	.3010	.7100	.9300	**1.0000**
Multi-objective 3	.5382	.5072	.7153	.7840	.7520	.7093	.7833	**.9987**	.6547	.7213	.5860	.6133	.9833	.1147	.9193	.4860	.9980
Rank	*12.0*	*5.9*	*8.4*	*8.2*	*5.9*	*12.0*	*8.4*	*6.0*	*6.0*	*2.6*	*8.9*	*15.0*	*4.1*	*16.0*	*9.4*	*10.0*	***1.6***

TABLE 3F. Comparison with 16 diverse algorithms on real-world data sets, measured by ACC.

Data sets/Methods	K-M++	SC	AC-A	AC-W	AC-S	AC-C	AC-CR	DPC	DPCLP	SNNDPC	DPA	FINCH	DB	AP	BP	RCC	TC
YTF	.5100	.2604	.2565	.5847	.2018	.3658	.2316	.6352	.1749	.4762	.4496	.5789	.5295	.2180	.5548	.6459	**.6973**
MNIST	.8277	.7008	.8332	NA	.7942	.8289	NA	.9733	NA	NA	.0903	.1580	.7014	.0308	.0550	.6091	**.9735**
COIL-100	.5424	.6368	.2794	.6053	.3504	.3521	.2175	.5447	.0135	.3089	.5385	.3922	.4313	.3542	.2532	.8307	**.8633**
Shuttle	.3932	.7581	.7834	.7663	.7862	.6386	.7837	.8893	NA	NA	.2912	.5788	.8230	.0216	.0669	.6619	**.9063**
RNA-seq	.8762	.9988	.3645	.9963	.3758	.7403	.3758	.7928	.3745	.9975	.9850	.8240	.5918	.3109	.9288	.9988	**.9988**
Haberman	.5842	.5229	.7320	.7288	.7386	.7320	.7353	.5425	.7059	.6863	.6209	.4771	.7320	.1438	.7386	.2222	**.7549**
Zoo	.7902	.7102	.8812	.8218	.5347	.8119	.8812	.7228	.3960	.8614	.4752	.9010	.7921	.6040	.4158	.8515	**.9109**
Atom	.7037	.5025	.6575	.6575	1.0000	.6450	.5250	.6963	1.0000	1.0000	.8163	.5675	1.0000	.1150	.6250	.5700	**1.0000**
Soybean	.9157	1.0000	.8298	1.0000	1.0000	1.0000	.8298	.9149	.8936	1.0000	.5745	.7872	.8298	.7660	.2128	1.0000	**1.0000**
Cell-track	.8870	.9000	.8750	.8750	.9000	.8750	.8750	.6000	.6250	.5250	.9000	.9000	**.9250**	.5750	.5500	.5000	.9000
CMU-PIE	.2328	.6524	.1961	.2272	.9496	.1719	.1457	.7892	.1089	.1499	.7539	.3592	.6933	.3617	.0525	.4097	**1.0000**
Reuters	.2571	.3364	.5271	.2675	.4611	.2395	.5402	.3286	.4518	.3193	.4325	.3331	**.5950**	.0617	.4707	.2997	.4054
Rank	*9.0*	*7.7*	*8.6*	*7.7*	*6.5*	*9.1*	*10.0*	*7.5*	*12.0*	*9.6*	*9.0*	*9.2*	*5.8*	*14.0*	*11.0*	*8.0*	***1.7***

ranked highest on AMI and ACC, achieving an NMI score of 86.04%.

For object recognition, the high-dimensional COIL-100 data set [47] was used, comprising 7,200 samples across 100 objects. TC achieved the best results across all indices, with an NMI of 97.2%.

The PIE recognition task used the CMU-PIE data set [48] with 2,856 frontal images under 43 lighting conditions. TC completed this task with 100% accuracy across all indices.

The biology domain included tasks in gene expression analysis, cell tracking, and animal recognition. For gene expression analysis, the RNA-seq data set [49] included gene expressions from patients with five tumor types. TC achieved top scores across all indices, correctly identifying all tumor types with an ACC of 99.88%.

In the cell tracking task using the Cell-track data set [50], TC achieved a competitive ACC score of 90% in distinguishing cells between the RGDS and FSL layers.

For animal recognition, the Zoo data set from the UCI Repository [51] was used, where TC achieved the best ACC score of 91.09%.

Table 4. Additional comparison with six latest density-based or automatic algorithms, measured by AMI.

Data sets/Methods	DPC-DVND	VDPC	KSFDPC	SMMP	CDC	DBDPC	TC
YTF	.6034	.3470	.8212	.6995	.5901	.7722	**.8384**
MNIST	.8645	.8877	.9023	.9292	.4333	.9041	**.9295**
COIL-100	.9205	.5575	.8652	.5457	.7011	.6238	**.9439**
Shuttle	.4580	.5679	.1950	.4011	.3906	.2755	**.6389**
RNA-seq	.9944	.0001	.6249	.9944	.8377	.7551	**.9944**
Haberman	.0164	-.0016	**.1073**	.0005	.0053	.0087	.0633
Zoo	.7031	.7191	.7674	.5315	.5972	.5841	**.8546**
Atom	.4520	.0619	1.0000	1.0000	.3400	.4113	**1.0000**
Soybean	1.0000	1.0000	1.0000	.0000	.7047	.3861	**1.0000**
Cell-track	.1555	.4696	.4696	.0000	**.6133**	.5360	.5360
CMU-PIE	.2740	.7084	.4798	.1160	.6778	.7367	**1.0000**
Reuters	.3236	.3643	.0797	.0035	.4100	.2877	**.4770**

In the medical domain, TC was tested on soybean disease diagnosis and breast cancer survival prediction. In the Soybean data set [52], TC accurately identified four diseases, achieving 100% across all indices.

In breast cancer survival prediction, Haberman's Survival data set [53] was used, categorizing survival times based on patient and surgery factors. TC identified three clusters with the best ACC score, closely matching the ground truth. An additional cluster represented samples with intermediate survival times.

In the physics domain, the Atom data set [54], featuring 3D data with two clusters, was used. TC achieved perfect scores across all indices, effectively distinguishing between the atom's kernel and hull.

For NLP, the Reuters data set [55] with 8,067 documents was used. TC correctly identified category counts with competitive accuracy.

Lastly, in the astronomy domain, TC was tested on the NASA Shuttle data set, with 58,000 observations across seven radiator subsystem conditions [1]. TC ranked highest across all indices, achieving an ACC of 90.63%.

3.4 Additional Comparison with Six Latest Density-based or Automatic Algorithms

Previously, for a comprehensive evaluation, we compared TC with 16 classical and recent clustering algorithms across various types. To add timeliness to our results, we now provide an additional comparison between TC and six of the latest algorithms on real-world data sets, including three density-based algorithms (DPC-DVND [56], VDPC [57], and KSFDPC [58]) and three automatic clustering algorithms (SMMP [59], CDC [46], and DBDPC [60]). Using AMI scores for performance evaluation, as shown in Table 4, TC outperformed these latest algorithms on 10 out of 12 real-world data sets, consistently demonstrating a performance advantage.

3.5 Result Analysis

3.5.1 Clustering Performance Analysis

According to the comprehensive quantitative comparison (Tables 3 and 4), TC demonstrated a significant performance advantage over a range of classical and recent algorithms. As shown in Table 3, TC ranked highest overall on both synthetic and real-world data sets. Notably, certain algorithms perform well on either synthetic or real-world data but struggle to excel on both. For instance, SNNDPC ranked 2.8 in AMI on synthetic data sets yet dropped to 10.0 on real-world data. On real-world data sets, RCC consistently ranked second only to TC in both the AMI and NMI indices. Across all indices, TC surpassed the next-best algorithm by approximately threefold in average ranking on real-world data.

From a clustering mechanism perspective, TC's constrained merging strategy based on 1-nearest clusters effectively identifies diverse shapes and structures compared to partition-based algorithms like K-M++ and SC. Compared to hierarchical algorithms such as AC-S and FINCH, TC's 1-nearest cluster constrained merging approach better captures true proximity and structural relationships within data, preventing potential merging errors. Additionally, TC's Torque Gap mechanism enables automatic determination of cluster numbers or granularity, a feature often lacking in most hierarchical algorithms. Unlike density-based methods that rely on local, point-level density, TC operates on a global, cluster-level mass, making it more robust to varying densities between clusters.

3.5.2 Automatic Cluster Number Determination Analysis

Accurately estimating the number of clusters is crucial for parameter-free or automatic clustering algorithms. In this part, we compared TC with the automatic algorithms referenced in sections 3.3 and 3.4, focusing on their average error in automatically determining cluster numbers across real-world data sets. Following standard practice, we used the mean error metric, defined as the average absolute difference between the cluster numbers determined by each algorithm and the ground-truth values across data sets. As shown in Table 5, TC achieved a lower mean error in estimating the number of clusters compared to nine other parameter-free or automatic clustering algorithms across 12 real-world data sets.

3.6 Comparison of Execution Time

To provide a clearer perspective on TC's efficiency, we compared its execution time with the nine advanced parameter-free or automatic clustering algorithms mentioned previously across 12 real-world data sets. All algorithms were implemented in MATLAB or Python, and the tests were conducted on a workstation equipped with two 14-core Intel Xeon 6132 CPUs (2.6 GHz base frequency, up to 3.7 GHz) and 96GB of RAM. As shown in Table 6, TC ranked as the second fastest parameter-free algorithm on average, just behind SMMP. However, as seen in Table 4, TC outperformed SMMP in terms of AMI scores across all 12 real-world data sets.

3.7 Extended Evaluation on 56 Additional Challenging Data Sets

We performed an additional comprehensive evaluation on a total of 56 data sets, which include 27 data sets with noise, outliers, overlaps, or other complex distributions; nine synthetic data sets with unbalanced clusters; five synthetic data sets with uniform noise; and 15 high-dimensional gene expression data sets characterized by poor separation. The results, as detailed in the supplement, indicate that TC consistently demonstrated a notable performance advantage across these challenging data sets.

Table 5. Mean error comparison in estimating the number of clusters with other automatic or parameter-free algorithms.

Data sets/Methods	#C	DPA	FINCH	DB	AP	BP	RCC	SMMP	CDC	DBDPC	TC
YTF	**40**	787	39	81	452	92	135	228	513	77	50
MNIST	**10**	441	143	7	538	701	47	**10**	1798	18	**10**
COIL-100	**100**	129	44	64	567	32	90	24	466	51	129
Shuttle	**7**	336	16	8	474	204	144	53	1696	36	4
RNA-seq	**5**	6	4	3	35	**5**	**5**	**5**	11	**5**	**5**
Haberman	**2**	5	3	**2**	18	1	10	28	15	6	3
Zoo	**7**	11	6	3	13	2	11	16	8	4	6
Atom	**2**	**2**	29	**2**	54	9	68	**2**	31	8	**2**
Soybean	**4**	2	3	3	6	1	**4**	1	3	2	**4**
Cell-track	**2**	**2**	**2**	1	4	**2**	20	1	**2**	**2**	**2**
CMU-PIE	**68**	61	271	67	231	3	41	2	187	47	**68**
Reuters	**30**	12	43	2	598	2	892	3	171	15	**30**
mean_error	-	130.92	37.17	9.83	226.08	93.08	105.33	36.83	385.50	14.50	**3.67**

Table 6. Execution time (in Seconds) comparison of TC and other automatic or parameter-free algorithms.

Data sets/Methods	DBDPC	DPA	FINCH	DB	AP	BP	RCC	CDC	SMMP	TC
YTF	634.698	20.875	30.861	372.299	390.800	5632.088	1662.751	8.528	3.372	33.308
MNIST	1286.914	183.370	43.987	27.488	19237.044	1476.800	189.323	127.108	1.948	45.387
COIL-100	652.329	76.623	95.246	1096.463	160.515	25889.481	7329.632	10.578	8.039	34.032
Shuttle	2240.385	282.977	20.379	24.106	9926.000	1085.206	306.054	106.689	2.691	21.490
RNA-seq	3.875	1.736	2.179	6.517	0.840	150.172	92.696	1.205	0.352	0.293
Haberman	0.035	0.072	0.546	0.002	0.225	0.120	0.116	0.362	0.012	0.068
Zoo	0.008	0.027	0.543	0.001	0.118	0.026	0.062	0.157	0.008	0.015
Soybean	0.010	0.012	0.611	0.001	0.041	0.010	0.035	0.091	0.003	0.007
Cell-track	0.001	0.011	0.533	0.001	0.042	0.008	0.032	0.074	0.001	0.007
CMU-PIE	3.105	3.798	1.105	3.941	18.003	84.499	34.618	3.509	0.113	1.821
Reuters	89.713	7.091	5.674	54.734	662.929	776.262	346.028	12.995	1.113	10.285
Mean	446.460	52.417	18.333	144.140	2763.300	3190.400	905.580	24.663	**1.604**	**13.338**

3.8 Comparison with Deep Clustering Algorithms on High-Dimensional Image Data Sets

Image data sets, with their inherently high dimensionality, often challenge traditional clustering algorithms, which may struggle to achieve good results on such data. Recent research has turned to deep neural networks to learn clustering-friendly low-dimensional representations, leading to significant improvements in clustering performance on image data sets. Accordingly, in this section, we compared TC with the latest deep clustering algorithms across several challenging image data sets, including UMIST [61], FRGC-v2.0 [62], COIL-20 [63], Pendigits [64], as well as COIL-100 [47] and CMU-PIE [65].

To offer a comprehensive comparison, we combined results from the leaderboard on "Papers with Code[1]," which ranked the performance of open-source algorithms, with findings from recent deep clustering research. Table 7 summarizes the results for the top six deep clustering algorithms across these image data sets, measured by NMI. TC demonstrated top performance on CMU-PIE and UMIST while performing competitively on other data sets. Although TC does not utilize deep representations, it achieved results comparable to those of deep clustering algorithms on these challenging data sets.

Deep clustering algorithms, however, face particular challenges, such as complex hyperparameter tuning, limited interpretability, and high computational costs. In high-dimensional data, significant variations in cluster density can complicate density estimation for each point, affecting the performance of density-based methods. TC, by contrast, bypasses point-level density estimation and instead relies on cluster-level mass estimation, where the mass of a cluster is defined as the number of points it contains. This approach enables TC to sustain robust performance in high-dimensional spaces, maintaining stable cluster mass even when density varies within natural clusters formed by constrained 1-nearest neighbor merging.

4 Discussion

4.1 Differences between TC and Other Hierarchical Clustering Algorithms

Even though TC is a hierarchy-based clustering algorithm, it introduces several key distinctions. Firstly, most hierarchical clustering methods rely entirely on unconstrained nearest-neighbor statistics. In contrast, TC incorporates a simple but effective constraint (as defined in Eq. (1)), which helps to avoid erroneous mergers (see Fig. 4). This concept is inspired by gravitational interactions observed in galaxy mergers. Secondly, at each stage of TC, if two neighboring clusters meet the requirement set by Eq. (1), a connection is formed, allowing for parallel mergers. Consequently, large clusters can form within only a few iterations, significantly boosting efficiency and reducing computation time (see Table 6). Traditional hierarchical clustering methods, however, generally require at least $n-K$ mergers to achieve

[1] https://paperswithcode.com/task/image-clustering

Table 7. Comparison with deep clustering algorithms on high-dimensional image data sets, measured by NMI.

Data sets	CMU-PIE	COIL-100	COIL-20	FRGC-v2.0	UMIST	Pendigits
Rank 1	1 JULE [48]	**.985** JULE [48]	**1** JULE [48]	**.651** DNB [66]	.917 DSCFEDL [67]	**.868** EAEDC [68]
Rank 2	1 DDSNnet [69]	.946 A-DSSC [70]	.981 DSC-FEDL [67]	.610 DEPICT [71]	.893 S^2DSCAG [72]	.863 N2D [73]
Rank 3	.970 DAutoED [74]	.943 J-DSSC [70]	.979 SADSC [75]	.580 MI-ADM [76]	.890 DSC-DAG [72]	.820 DnC-SC [77]
Rank 4	.965 MI-ADM [76]	.910 DGMM [78]	.974 S^2DSCAG [72]	.574 JULE [48]	.881 RGRL [79]	.817 DipDECK [80]
Rank 5	.964 DEPICT [71]	.905 DBC [81]	.958 DSC-DAG [72]	.544 DPSC [82]	.877 JULE [48]	.814 GCML [83]
Rank 6	.925 DPSC [82]	.886 DDSNnet [69]	.910 DGMM [78]	.522 DDSNnet [69]	.851 DNB [66]	.801 AESC [84]
TC	**1**	.972	.960	.627	**.931**	.837

K clusters. Thirdly, TC determines the optimal number of clusters automatically by eliminating abnormal connections using a novel *TGap* metric, whereas most existing hierarchical algorithms still require manual cluster number selection or granularity adjustment. Lastly, TC demonstrates robustness to noise and outliers and is capable of isolating noise clusters, which is a challenge for many classical agglomerative clustering methods. As illustrated in the case study in Fig. 4 and supported by the empirical results in Table 3, TC outperforms other hierarchy-based clustering algorithms.

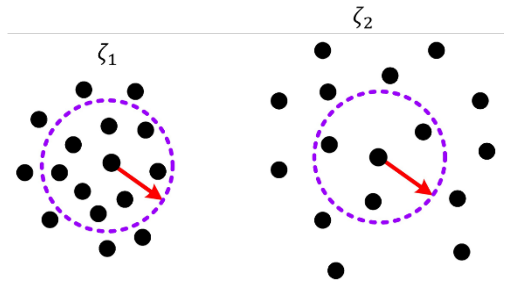


Figure 6. Example of a varying-density data set. TC treats clusters ζ_1 and ζ_2 as having the same mass (16 points) despite their differing local densities.

4.2 Differences between TC and Density Peak Clustering (DPC) Algorithms

Although TC and DPC share similarities in leveraging proximity relationships and quantitative metrics (e.g., mass or density) to delineate cluster structures, they exhibit significant differences in their methodologies and robustness. DPC defines density at the point level using a manually adjusted, fixed cutoff distance, making it highly sensitive to density variations across the data set. In contrast, TC eliminates the need for cutoff distance adjustments by relying on cluster-level mass estimation (i.e., the number of points in each cluster), ensuring stability and robustness in characterizing cluster structures. Regardless of density variations or high-dimensional sparsity, clusters formed during TC's merging process maintain the same mass if they contain the same number of points. For example, as illustrated in Figure 6, while the local density in cluster ζ_1 is significantly higher than that in ζ_2 under the same cutoff distance, TC treats both ζ_1 and ζ_2 as having the same mass (16 points). More broadly, TC avoids the need for density estimation required by most density-based algorithms, such as DPC, by intuitively using mass as the metric to characterize clusters. This approach not only circumvents the challenges of accurately estimating density in high-dimensional spaces caused by the curse of dimensionality, thereby improving performance, but also eliminates the need for density-related hyperparameters like cutoff distance, offering a fully parameter-free framework. Furthermore, in terms of clustering perspective, TC dynamically estimates mass at the cluster level throughout the merging process, progressively reflecting a global perspective that adapts to the overall structure of complex data sets. Conversely, DPC relies on fixed local density estimates, inherently lacking a global perspective, which limits its effectiveness in handling data sets with irregular or diverse distributions. Finally, regarding label assignment mechanisms, TC begins by treating each point as an individual cluster and iteratively merges clusters with their nearest neighbor of higher mass, resulting in a hierarchical clustering tree. It then identifies abnormal connections and prunes the tree to determine the final clusters, where the merging process serves as label assignment, and the pruning process acts as error correction. This "assign-first, error-correct-later" strategy mitigates potential errors during label assignment, leading to greater robustness on complex data sets. In contrast, DPC selects cluster centers based on high-density points and assigns labels by connecting each point to its nearest neighbor of higher density. Errors in cluster center selection, caused by density variability or inaccuracies in density estimation (particularly in high-dimensional spaces), can propagate throughout the label assignment process, as DPC lacks a corrective mechanism to address such errors.

4.3 Potential limitations of TC

On one hand, because TC relies on global mean values (i.e., mean_M, mean_D, and mean_$\frac{D}{M}$) to detect cluster halos, it may not be able to precisely identify all instances of non-uniform noise. On the other hand, since TC assumes abnormal connections have relatively large distance values when identifying them, TC may face challenges with data sets containing an extremely large number of clusters lacking clear boundaries. This limitation is a shared challenge among most existing clustering algorithms.

5 Conclusion

In this paper, we introduce an effective clustering algorithm that serves as a viable alternative to existing methods, capable of identifying clusters with varying shapes, sizes, and densities. This algorithm is fully parameter-free, computationally efficient, and robust to noise and outliers. TC operates on a straightforward, intuitive principle: a cluster should merge with its nearest neighbor with higher mass unless both clusters have relatively large masses and a significant distance between them. By leveraging peaks in mass and distance, the algorithm effectively detects and removes incorrect mergers. In the experimental section, we conduct extensive tests to demonstrate TC's effectiveness and robustness, revealing a significant performance advantage over existing parameter-free, automatic, and other related clustering algorithms.

6 Future Work

In future work, we plan to explore two main enhancements for TC. First, when determining cluster halos, we will experiment with setting distinct thresholds for individual clusters instead of relying solely on global mean values, aiming to increase TC's robustness against non-uniform noise. Secondly, for data sets with a large number of clusters that lack clear boundaries, we plan to develop an extended version of TC, called Deep TC Clustering, by integrating neural network-based representation learning to obtain more discriminative representations and further enhance performance.

References

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

[2] A. Saxena *et al.*, "A review of clustering techniques and developments," *Neurocomputing*, vol. 267, pp. 664–681, Dec. 2017.

[3] 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.

[4] 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.

[5] S. Kpotufe and U. von Luxburg, "Pruning nearest neighbor cluster trees," *ArXiv11050540 Cs Stat*, May 2011, Accessed: Nov. 23, 2020.

[6] 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.

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

[8] 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*, Morgan Kaufmann, 2000, pp. 727–734.

[9] 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*, in KDD '04. New York, NY, USA: Association for Computing Machinery, Aug. 2004, pp. 551–556.

[10] 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.

[11] 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.

[12] 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.

[13] A. Lotfi, P. Moradi, and H. Beigy, "Density peaks clustering based on density backbone and fuzzy neighborhood," *Pattern Recognit.*, vol. 107, pp. 107449, Nov. 2020.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] O. Bertolami, F. Gil Pedro, and M. Le Delliou, "Dark energy–dark matter interaction and putative violation of the equivalence principle from the Abell cluster A586," *Phys. Lett. B*, vol. 654, no. 5, pp. 165–169, Oct. 2007.

[24] 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.

[25] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, Jun. 2014.

[26] 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.

[27] 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.

[28] S. C. Johnson, "Hierarchical clustering schemes," *Psychometrika*, vol. 32, no. 3, pp. 241–254, Sep. 1967.

[29] 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.

[30] 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, Jun. 2018.

[31] 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, Jun.

2021.

[32] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "Density-based spatial clustering of applications with noise," *Int. Conf. Knowledge Discovery and Data Mining*, 1996.

[33] B. J. Frey and D. Dueck, "Clustering by Passing Messages Between Data Points," *Science*, vol. 315, no. 5814, pp. 972–976, Feb. 2007.

[34] H. Averbuch-Elor, N. Bar, and D. Cohen-Or, "Border-Peeling Clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 7, pp. 1791–1797, Jul. 2020.

[35] 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.

[36] N. X. Vinh, J. Epps, and J. Bailey, "Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance," *J. Mach. Learn. Res.*, vol. 11, no. 95, pp. 2837–2854, 2010.

[37] P. Fränti and O. Virmajoki, "Iterative shrinking method for clustering problems," *Pattern Recognit.*, vol. 39, no. 5, pp. 761–775, May 2006.

[38] 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.

[39] H. Chang and D.-Y. Yeung, "Robust path-based spectral clustering," *Pattern Recognit.*, vol. 41, no. 1, pp. 191–203, Jan. 2008.

[40] 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.

[41] 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.

[42] 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.

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

[44] 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.

[45] 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.

[46] D. Peng *et al.*, "Clustering by measuring local direction centrality for data with heterogeneous density and weak connectivity," *Nat. Commun.*, vol. 13, no. 1, p. 5455, Sep. 2022.

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

[48] 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.

[49] 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.

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

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

[52] Tan M., Eshelman L., "Using Weighted Networks to Represent Classification Knowledge in Noisy Domains," *Mach. Learn. Proc. 1988*, pp. 121–134, Jan. 1988.

[53] S. J. Haberman, "Generalized residuals for log-linear models," *In Proceedings of the 9th international biometrics conference*, 1976, pp. 104–122.

[54] M. C. Thrun and A. Ultsch, "Clustering benchmark datasets exploiting the fundamental clustering problems," *Data Brief*, vol. 30, p. 105501, Jun. 2020.

[55] D. Cai, X. He, and J. Han, "Document clustering using locality preserving indexing," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 12, pp. 1624–1637, 2005.

[56] W. Zang *et al.*, "Density peaks clustering based on density voting and neighborhood diffusion," *Inf. Sci.*, vol. 681, p. 121209, Oct. 2024.

[57] Y. Wang, D. Wang, Y. Zhou, X. Zhang, and C. Quek, "VDPC: Variational density peak clustering algorithm," *Inf. Sci.*, vol. 621, pp. 627–651, Apr. 2023.

[58] C. Li, S. Ding, X. Xu, H. Hou, and L. Ding, "Fast density peaks clustering algorithm based on improved mutual K-nearest-neighbor and sub-cluster merging," *Inf. Sci.*, vol. 647, p. 119470, Nov. 2023.

[59] J. Guan, S. Li, X. He, J. Zhu, J. Chen, and P. Si, "SMMP: A Stable-Membership-Based Auto-Tuning Multi-Peak Clustering Algorithm," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 5, pp. 6307–6319, May 2023.

[60] J. Hou, H. Lin, H. Yuan, and M. Pelillo, "Flexible density peak clustering for real-world data," *Pattern Recognit.*, vol. 156, p. 110772, Dec. 2024.

[61] 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., in NATO ASI Series, Berlin, Heidelberg: Springer, 1998, pp. 446–456.

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

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

[64] 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, pp. 637–640 vol.2.

[65] 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.

[66] 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.

[67] 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, Sep. 2020.

[68] X. Huang, Z. Hu, and L. Lin, "Deep clustering based on embedded auto-encoder," *Soft Comput.*, 27(2), pp.1075-1090, 2021.

[69] 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, Sep. 2021.

[70] D. Lim, R. Vidal, and B. Haeffele, "Doubly Stochastic Subspace Clustering," arXiv:2011.14859. 2020 Nov 30.

[71] K. G. Dizaji, A. Herandi, C. Deng, W. Cai, and H. Huang, "Deep Clustering via Joint Convolutional Autoencoder Embedding and Relative Entropy Minimization," *2017 IEEE International Conference on Computer Vision (ICCV)*, IEEE Computer Society, Oct. 2017, pp. 5747–5756.

[72] 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.

[73] 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.

[74] M. Yang and S. Xu, "Orthogonal Nonnegative Matrix Factorization using a novel deep Autoencoder Network," *Knowl.-Based Syst.*, vol. 227, p. 107236, Sep. 2021.

[75] 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.

[76] 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.

[77] H. Li, X. Ye, A. Imakura, and T. Sakurai, "Divide-and-conquer based Large-Scale Spectral Clustering," *Neurocomputing*, 501, pp.664-678, 2022.

[78] J. Wang and J. Jiang, "Unsupervised deep clustering via adaptive GMM modeling and optimization," *Neurocomputing*, vol. 433, pp. 199–211, Apr. 2021.

[79] Z. Kang, X. Lu, J. Liang, K. Bai, and Z. Xu, "Relation-Guided Representation Learning," *Neural Netw.*, vol. 131, pp. 93–102, Nov. 2020.

[80] 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.

[81] F. Li, H. Qiao, and B. Zhang, "Discriminatively boosted image clustering with fully convolutional auto-encoders," *Pattern Recognit.*, vol. 83, pp. 161–173, Nov. 2018.

[82] 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, Aug. 2021.

[83] L. Wu, Z. Liu, J. Xia, Z. Zang, S. Li, and S. Z. Li, "Generalized Clustering and Multi-Manifold Learning With Geometric Structure Preservation," *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022, pp. 139–147.

[84] 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.

Jie Yang (M'23) received the Ph.D. degree from Australian Artificial Intelligence Institute (AAII), University of Technology Sydney (UTS). He currently is a postdoctoral fellow at Computational Intelligence and Brain-Computer Interface Lab, AAII, UTS, Australia. His current research interests include unsupervised learning, clustering, representation learning, and EEG data processing. In addition, he serves as a reviewer for many top-tier journals, such as IEEE Transactions on Knowledge and Data Engineering, IEEE Transactions on Fuzzy Systems, IEEE Transactions on Neural Networks and Learning Systems, IEEE Transactions on Industrial Informatics, and Information Sciences.

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); Chair of IEEE CIS Awards Committee (2022, 2023); 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 425 journal papers including about 200 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 40,000 times. Currently, his h-index is 96, and his i10-index is 466.