Toward Autonomous Distributed Clustering

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

***Abstract*—Compared to traditional centralized clustering, distributed clustering offers the advantage of parallel processing of data from different sites, enhancing the efficiency of clustering while preserving the privacy of the data at each site. However, most existing distributed clustering techniques require manual tuning of several parameters or hyperparameters, which can pose challenges for practical applications. This paper introduces a novel parameter-free distributed clustering framework known as distributed torque clustering (DTC). When dealing with data or subdata distributed across various sites, DTC predominantly executes two steps. The first step is a data reduction at each site using torque clustering, and the second step involves performing global clustering with weighted torque clustering. We compare DTC against six state-of-the-art distributed clustering algorithms and automatic centralized clustering techniques on ten large-scale or medium-scale datasets. The results show that the average rank of DTC is at least three times better than those of the other algorithms across all the datasets. Additionally, DTC can accurately predict the ground-truth number of clusters in nine out of ten datasets, further demonstrating its competitive performance and practical potential.**

***Index Terms*—Clustering, distributed clustering, parameter-free, autonomous, unsupervised learning.**

NOMENCLATURE

Symbol	Meaning
ζ_i, ζ_i^N	i-th cluster and its 1-nearest cluster.
$mass(\zeta_i)$, $mass(\zeta_i^N)$	Size of ζ_i or ζ_i^N.
M_i	Product of $mass(\zeta_i)$ and $mass(\zeta_i^N)$.
D_i	Square of the distance between ζ_i and ζ_i^N.
τ_i	Product of M_i and D_i.
$TGap_j$	Weighted reciprocal ratio of adjacent elements in the sorted array $\{\tau_j\}$.
$TC\langle Y_k\rangle$	Performing Torque Clustering (TC) on the data at site Y_k.
RS_k	Representative samples obtained at k-th site.
$\overline{\zeta_\beta}$	Mean vector of the samples in β-th cluster.
W_{RS_k}	Weights of the representative samples at k-th site.

I. INTRODUCTION

Distributed clustering is a data clustering technique that processes and analyzes datasets partitioned across multiple sites or machines, aiming to discover meaningful patterns and relationships while reducing communication overhead and computational complexity. Over the past few decades, numerous distributed clustering methods have been proposed. However, these methods often require users to preset various parameters or hyperparameters, significantly reducing their usability in practice. For example, Pedro et al. introduced two popular distributed clustering algorithms, including distributed K-means (DK-means) and distributed expectation-maximization (DEM) [1]. Although these two distributed algorithms exhibit improved robustness to initialization compared to their centralized counterparts, i.e., K-means [2] and EM, they still inherit the hyperparameters of K-means and EM, namely, the number of clusters. Ye et al., building on the idea of DK-means, proposed the distributed fuzzy C-means (DFCM) method, which provides a soft assignment for the data samples in each site, i.e., a membership for each category [3]. However, this algorithm still requires a manual specification of the cluster number, and its performance is sensitive to the hyperparameter *m* (the degree of fuzziness). Eshref et al. combined the DBSCAN algorithm with distributed learning paradigms to propose the density-based distributed clustering (DBDC) algorithm [4]. This algorithm consists of two main processes. First a local clustering step is used to extract a small number of representative data samples from each site; second, representative samples are gathered into a hub to complete the global clustering step, and then the labels are propagated back to each site to complete the full clustering. The DBDC algorithm inherits the two manually specified hyperparameters of DBSCAN [5], Eps and MinPts. Maria et al. proposed two new center-based distributed clustering methods from a topological graph perspective, including distributed K-means and distributed K-median [6]. Compared to previous DK-means, these two algorithms have lower communication costs. However, these two new paradigms introduce more hyperparameters. In addition to specifying the cluster number, the sampling parameter *t* and the number of neighbors for each node in the topological graph also requires user input. This poses significant difficulties for users applying these two algorithms in real-world scenarios. Malika et al. proposed a distributed dynamic clustering algorithm (D2CA) that combines K-means and agglomerative clustering methods [7]. D2CA uses K-means for data reduction at each site and then completes clustering in the hub by continuously merging the subclusters. Although D2CA can automatically determine the number of clusters, it requires a specification of the number of subclusters at each site to obtain representative samples. Matthias et al. proposed a distributed clustering algorithm based on sampling local density estimates [8]. This algorithm is sensitive to a hyperparameter called the window width *h* when estimating density, significantly reducing the algorithm's usability. D.K. Tasoulis et al., inspired by grid-based clustering, proposed a distributed clustering algorithm called K-windows [9]. This algorithm inherits the drawbacks of grid-based clustering, requiring at least seven interval-related hyperparameters to be set. Additionally, the algorithm is only suitable for low-dimensional datasets.

Compared to distributed clustering, centralized clustering methods were introduced earlier for clustering centralized datasets. One of the most typical and classic centralized clustering algorithms is K-means [2], which requires all the data samples in the dataset to be concentrated

Manuscript received 9 January 2024; accepted 11 February 2024. Date of publication 2 April 2024; date of current version 27 March 2025. This work was supported in part by the Australian Research Council (ARC) under discovery under Grant DP210101093 and Grant DP220100803, in part by the Australian National Health and Medical Research Council (NHMRC) Ideas under Grant APP2021183, in part by the UTS Human-Centric AI Centre funding sponsored by GrapheneX (2023-2031), in part by the Australia Defence Innovation Hub under Grant P18-650825, and in part by Australian Cooperative Research Centres Projects (CRC-P) Round 11 under Grant CRCPXI000007. *(Corresponding authors: Jie Yang; Chin-Teng Lin.)*

The authors are with the Computational Intelligence and Brain Computer Interface Lab, GrapheneX-UTS HAI Centre, Australian AI Institute, SoCS, FEIT, University of Technology Sydney, Australia (e-mail: jie.yang-1@uts.edu.au; chin-teng.lin@uts.edu.au).

Recommended for acceptance by J. Catalão.
Digital Object Identifier 10.1109/TETCI.2024.3378603

in a single site. Moreover, although the field of distributed clustering has not yet developed parameter-free algorithms, some parameter-free algorithms have already been developed in the centralized clustering domain. For instance, M. Saquib Sarfraz et al. proposed a parameter-free hierarchical clustering algorithm called FINCH, which is based on the 1-nearest neighbor chain theory and provides multiresolution partitions [10]. With only the input of the dataset, FINCH can automatically determine the number of clusters and their corresponding multiresolution partitions. Sohil Atul Shah et al. introduced a robust continuous clustering (RCC) algorithm, which is based on the mutual K-NN graph and a novel continuous objective [11]. RCC can automatically learn clustering-favorable representations and the proper number of clusters without the need for specifying any hyperparameters. Hadar Averbuch-Elor et al. proposed a nonparametric density-based clustering algorithm called border peeling (BP) clustering [12]. BP is based on the idea that each latent cluster consists of layers surrounding its core, with the outer layers or border points implicitly separating the clusters. In each iteration, BP uses fixed parameter settings to complete the labeling assignment of the remaining samples, such as $K = 20$ for K-NN queries. Maria d'Errico et al. introduced a parameter-free automatic density peak clustering (DPA) algorithm [13]. DPA mainly addresses two issues present in the original density peak clustering (DPC) [14]. First, DPC still requires a manual determination of the number of cluster centers; second, DPC's performance on high-dimensional datasets is unsatisfactory. Furthermore, DPA employs a parameter-free density estimator to calculate the density of each data sample. Jian Hou et al. combined dominant set (Dset) clustering and DPC to propose a sequential clustering algorithm for handling Gaussian distribution clusters [15]. This algorithm first extracts the initial clusters based on the DSet algorithm and then uses an improved DPC-based method to expand the initial clusters to obtain the final clusters. Additionally, this sequential algorithm proposes a heuristic-based adaptive method to automatically determine the cutoff distance in the DPC algorithm, making the entire procedure parameter-free. Most of the parameter-free centralized clustering algorithms introduced above are based on heuristic strategies. However, there are also uncertainty and optimization-based clustering algorithms that possess parameter-free characteristics [33], [34], [35], [36]. For example, Zhaoyin Shi et al. proposed a parameter-free robust ensemble framework for fuzzy clustering that addresses challenges such as misaligned membership matrices and outlier effects by cascading membership matrices, mining latent spectral matrices, and introducing a robust weighted mechanism [33]. This results in a model that improves performance, stability, and applicability without the need for hyperparameter tuning. Xuelong Li et al. proposed a scalable and parameter-free graph fusion clustering framework, optimized using the block-coordinate descent method to efficiently minimize the loss function and obtain the optimal solution [35]. Although the centralized clustering methods described above are parameter-free, they cannot handle data samples that exist in distributed sites. In addition, requiring all the samples to be concentrated in a single site make these centralized parameter-free clustering methods consume considerable amounts of local memory and do not protect the privacy of each site.

The literature review above reveals that numerous parameter-free algorithms have been developed within the centralized clustering field to enhance their applicability in real-world situations. However, there is currently no parameter-free distributed clustering method available. This paper introduces a novel parameter-free distributed framework called distributed torque clustering (DTC) to address this absence. DTC is built upon our previously proposed torque clustering (TC), an autonomous, nonparametric centralized clustering method that exhibits a commendable performance on datasets containing either convex or nonconvex clusters [16]. Unlike the existing distributed clustering methods, DTC does not require the prespecification of any hyperparameters or parameters, such as the number of clusters or a density threshold. Additionally, DTC surpasses the traditional parameter-free centralized clustering methods by processing data across multiple sites and boosting clustering efficiency through parallelization. Furthermore, DTC maintains other distributed clustering features, including effective data privacy protection for each site or machine. The primary research contributions of this paper are as follows:

1) We propose a novel parameter-free distributed clustering framework, namely, distributed torque clustering (DTC).
2) We propose a method called weighted torque clustering (WTC) to enhance the robustness of DTC during the global clustering phase, particularly against sparse data distribution.
3) We compare DTC with six state-of-the-art distributed clustering algorithms or automatic centralized clustering techniques across ten large-scale or medium-scale datasets, demonstrating the efficacy and feasibility of DTC.

The subsequent sections of the paper are structured as follows: in the method section, it briefly introduces Torque Clustering (TC), an innovative technique that clusters data based on mass and distance, and extends this method to data reduction and global clustering via Weighted Torque Clustering (WTC). The framework of DTC is then delineated, highlighting its efficient three-step process: data preparation at sites, local clustering using TC, and global clustering using WTC. The experiments section describes the datasets used, both synthetic and real-world, and outlines the experimental setup for benchmarking DTC against other clustering methods. Results and analysis showcase DTC's superior performance across various datasets in comparison to existing methods. The paper further explores the automatic determination of cluster numbers, the impact of WTC, DTC's robustness in different scenarios, and its application in cell type identification using scRNA-seq datasets. Finally, the conclusion section summarizes DTC's effectiveness in distributed clustering, emphasizing its parameter-free nature, accuracy, and practicality in handling large-scale datasets.

II. Method

A. Torque Clustering

In our earlier study, we proposed a centralized hierarchical clustering approach without parameter tuning, named torque clustering (TC) [16], which is based on the cluster mass or size. The concept of TC primarily consists of two stages. First, larger clusters guide smaller ones through the merging process, effectively minimizing the inaccurate merging commonly found in traditional hierarchical clustering. This procedure is referred to as mass-constrained merging within a TC. Second, the combination of clusters or subclusters generates a hierarchical clustering tree. Subsequently, TC autonomously trims the clustering tree by considering the mass and distance peaks, resulting in a partition with an appropriate number of clusters.

Given a dataset X, initially, each sample is its own cluster. Given the number of samples contained in a cluster as the mass of the cluster (i.e., the cluster size), therefore, in the beginning, the mass of each cluster equals one. The following rule is then applied to form connections between the clusters:

$$\zeta_i \rightarrow \zeta_i^N,\ if\ mass\left(\zeta_i\right) \leq mass\left(\zeta_i^N\right) \tag{1}$$

where ζ_i denotes the *i*-th cluster and ζ_i^N denotes the 1-nearest cluster of ζ_i. $mass(\zeta_i)$ represents the mass of ζ_i (i.e., the size of ζ_i). Similarly, $mass(\zeta_i^N)$ is the mass of ζ_i^N. The symbol "$\rightarrow$" denotes a connection (i.e., merger) C_i between ζ_i and ζ_i^N. This process can be defined in a graph *G, and* new clusters can be obtained by calculating the connected

components of G. Upon completing one iteration, the merger process is repeated according to (1) until all the clusters eventually combine into a single cluster, forming a hierarchical tree. Each level of the hierarchical tree can be viewed as a partition with a distinct level of granularity.

Each connection (i.e., merger) C_i has two intuitive properties. One is the product of the mass of the two clusters it connects

$$M_i = mass\left(\zeta_i\right) \times mass\left(\zeta_i^N\right) \tag{2}$$

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

$$D_i = d^2\ \left(\zeta_i, \zeta_i^N\right) \tag{3}$$

An appropriate partition can be derived from a specific layer (granularity) within the clustering tree. In addition, TC leverages an index called the Torque Gap (T-Gap) to automatically prune the hierarchical clustering tree to obtain the partition with the total proper number of clusters. TC first calculates the torque value τ_i of each connection.

$$\tau_i = M_i \times D_i \tag{4}$$

Subsequently, TC arranges all the connections in a descending sequence based on their respective torque values, creating what is known as the torque sorted connections list (TSCL). On the TSCL, the TC calculates the T-Gap between each connection and its next connection as

$$TGap_j = \omega_j \frac{\grave{\tau}_j}{\grave{\tau}_{j+1}}, \grave{\tau}_{i+1} \neq 0 \tag{5}$$

where $\grave{\tau}_j$ denotes the torque value of each connection in the TSCL. ω_j is a weight related to the proportion of connections among the top j connections of TSCL that have relatively large M_j, D_j and τ_j values. TC also provides specific mechanisms to automatically determine it [16]. After finding the largest T-Gap, i.e., $TGap^*$, on TSCL, those connections are removed before $TGap^*$ to complete the pruning of the clustering tree and to obtain the final partition with the proper cluster number.

B. Data Reduction Using Torque Clustering for Each Site

To minimize computational costs, most existing distributed clustering methods require the extraction of the representative samples from each site's subdataset. This is typically accomplished using clustering algorithms. For instance, [1] employs K-means to individually cluster data at each site, using the cluster centers as representative samples. [3] utilizes FCM for clustering data at each site to obtain the representative samples, while another reference [4] relies on DBSCAN to cluster the data from each site. Compared to the first two methods, the latter approach [4] more accurately identifies the data manifold structures, as DBSCAN [5] can adapt to datasets with intricate shapes. Nonetheless, these clustering algorithms necessitate the manual presetting of parameters or hyperparameters. For instance, both [1] and [3] require specifying the number of clusters, while [4] demands the configuration of DBSCAN's two hyperparameters, Minpts and Eps. This introduces a considerable inconvenience to the process of extracting representative samples using clustering.

In contrast to previous approaches, we employ TC to cluster the data at each site for the extraction of the representative samples. Let the subdata in each site be Y_k. We use TC to perform clustering on Y_k to automatically obtain r_k clusters in each site, and denote this step as

$$TC\left\langle Y_k\right\rangle = \left\{\zeta_1, \zeta_2, \ldots, \zeta_{r_k}\right\}, \tag{6}$$

where $TC\langle\rangle$ means the procedure of TC. Furthermore, we take the mean vector of the samples in each cluster of each site as the representative, that is,

$$RS_k = \left\{\overline{\zeta_1}, \overline{\zeta_2}, \ldots, \overline{\zeta_{r_k}}\right\}, \tag{7}$$

Algorithm 1: Pseudocode of the Proposed DTC.

Input: subdata from each site $\{Y_k\}_{k=1}^S$.
Output: final partition P.
// Data reduction using TC
parallel for k=*1:S* **do**
Perform Torque Clustering (TC) on Y_k in each site to obtain r_k clusters, i.e., $\{\zeta_\alpha\}_{\alpha=1}^{r_k}$, by (1)–(6).
Compute representatives RS_k, i.e., $\{\overline{\zeta_\beta}\}_{\beta=1}^{r_k}$, by (7).
Record correspondence between each sample x and RS_k, i.e., $x \rightarrow RS_k$.
Compute weights of representatives W_{RS_k}, i.e., $\{mass(\zeta_\gamma)\}_{\gamma=1}^{r_k}$, by (8).
end
// Global clustering using WTC
Perform TC with initial sample weights (i.e., Weighted TC) based on whole representatives $\{RS_k\}_{k=1}^S$ and corresponding weights $\{W_{RS_k}\}_{k=1}^S$ to obtain partition P^* for the whole representatives in the hub by (1)–(5).
Compute final partition P according to P^* and $\{x \rightarrow RS_k\}_{k=1}^S$

where RS_k means the representative samples obtained at each site.

Utilizing TC for extracting representatives from each site offers two primary advantages compared to previous methods. First, TC eliminates the need for a manual hyperparameter adjustment and can automatically determine the number of clusters, which corresponds to the number of representatives. Second, in contrast to DBSCAN, TC is capable of recognizing, not only datasets with intricate shapes but also the clusters with diverse modalities, such as those with varying densities [16].

C. Global Clustering Using Weighted Torque Clustering

Upon extracting representatives from each site, they are forwarded to a hub for the global clustering phase. Unlike most previous methods, which treat these representatives as new data samples and directly perform clustering on them, our approach accounts for the characteristics of each representative's domain. For instance, some representatives may originate from high-density areas, while others come from low-density regions. As a result, we enhance the original TC to better represent the weight of each representative.

In (7), we obtain representatives from each site using TC. Here, we use the mass of the corresponding cluster for each representative as the weight of that representative, that is,

$$W_{RS_k} = \left\{mass\left(\zeta_1\right), mass\left(\zeta_2\right), \ldots, mass\left(\zeta_{r_k}\right)\right\}, \tag{8}$$

where W_{RS_k} means the weights of the representative samples at each site, r_k denotes the number of representatives at this site. For example, for a representative $\overline{\zeta_1}$ and its corresponding cluster ζ_1, the weight of representative $\overline{\zeta_1}$ is mass (ζ_1), that is, the number of samples contained in cluster ζ_1.

At the hub, we consider representatives from all the sites along with their corresponding weights as a dataset to be processed. Subsequently, we execute weighted torque clustering (WTC) on this dataset to accomplish the global clustering phase. As illustrated in Fig. 1, we use a simple example to elucidate the differences between TC and WTC. Several shapes represent the representatives obtained from all the sites. The type of shape also reflects their category in the ground truth. In the TC procedure on the left, each representative is either unweighted or has equal weights (i.e., all are equal to 1). After performing TC on these representatives, the automatically determined number of clusters

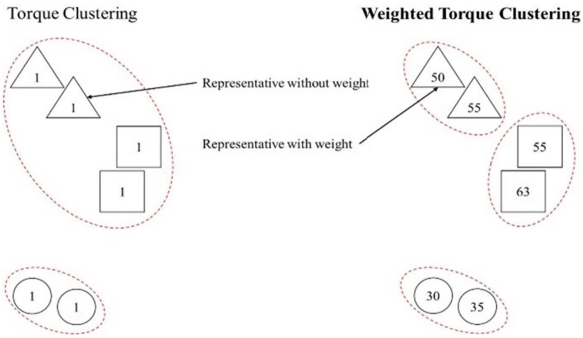

Fig. 1. Comparison between torque clustering and weighted torque clustering.

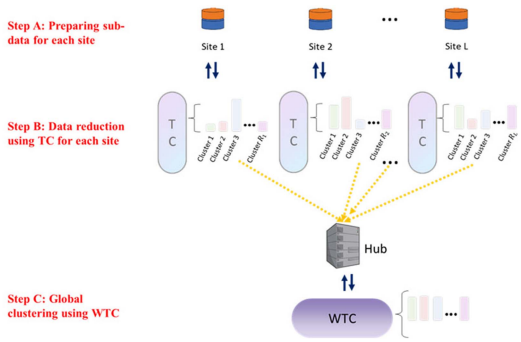

Fig. 2. Framework of distributed torque clustering (DTC).

is more likely to be 2, as indicated by the red dashed line. In the WTC procedure on the right, each representative carries a weight, which corresponds to the mass (or size) of the original cluster. After executing WTC on these representatives, the automatically determined number of clusters is more likely to be 3, as shown by the red dashed line. This is because, even though the triangle and rectangle categories are very close, both have a high weighted mass, resulting in the automatic trimming of the connection between them. The sole distinction between WTC and TC lies in the method for calculating the cluster mass. In TC, the mass of each cluster is determined by the number of samples it contains. Conversely, in WTC, the weighted mass of each cluster is calculated as the sum of the weights of the samples it encompasses.

Applying WTC for global clustering of all the representatives in the hub offers two main advantages. First, compared to previous distributed clustering methods, WTC eliminates the need for a manual adjustment of any hyperparameters, and can automatically determine the number of clusters, providing users with significant convenience. Second, on some relatively sparse datasets, after performing the data reduction step, the distribution of representatives will become sparser. If TC is used to perform global clustering on the representatives, the resulting cluster count may be less accurate, impacting the precision. However, using WTC can alleviate this issue to some extent.

D. Framework of Distributed Torque Clustering

In this section, we introduce the primary framework for distributed torque clustering (DTC). As depicted in Fig. 2, the DTC process entails three steps. In Step A, each site is required to prepare its subdata. Alternatively, one may divide the total data into multiple subdata segments. Step B involves the application of TC to each site individually, facilitating the formation of several clusters at each site, as represented by (6). Notably, the number of clusters obtained at each site is determined autonomously by TC based on the unique characteristics of the subdata. Furthermore, DTC computes the mean vector of the samples in each cluster of each site (i.e., (7)), using it as the representative, while also documenting the weight of each representative (i.e., (8)), which corresponds to the size of its cluster. Last, DTC transmits all the representatives along with their respective weights to the hub. In Step C, DTC employs WTC on all weighted representatives for global clustering, thereby yielding the final results. Furthermore, the users can assign labels to the subdata within each site using the labels of the representatives obtained from the global clustering, as well as the affiliation between the representatives and the sites.

DTC holds two significant advantages over previous distributed clustering algorithms. First, DTC involves two main steps, data reduction using clustering (Step B) and global clustering (Step C), both of which utilize either TC or its variant, WTC. Both TC and WTC can automatically determine the number of clusters without the necessity for additional tuning of the hyperparameters. This feature greatly enhances the user experience by offering increased convenience. Second, the computational complexity of TC or WTC, i.e., $O(n^2)$ can be decreased to $O(nlogn)$ when certain nearest neighbor approximation algorithms are used, such as k-d tree or local hashing [16]. Consequently, DTC also exhibits superior scalability.

Algorithm 1 presents the pseudocode of the proposed DTC. Below, we conduct a detailed analysis of the time complexity of the DTC based on Algorithm 1. We first analyze the complexity within the loop of steps 3–8. Step 4 requires a complexity of $O(|Y_k|^2)$, and each of steps 5–7 demands a complexity of $O(r_k)$. Furthermore, because $r_k \ll |Y_k|^2$, the total complexity of steps 4–7 is $O(|Y_k|^2)$. However, considering that steps 4–7 are executed in parallel at each site, the overall complexity of the loop in steps 3–8 is $\max_k O(|Y_k|^2)$. Step 9 requires a complexity of $O(\sum_{k=1}^{S} {r_k}^2)$, while step 10 demands a complexity of $O(\sum_{k=1}^{S} r_k)$. In summary, the total time complexity of the DTC is $\max_k O(|Y_k|^2) + O(\sum_{k=1}^{S} {r_k}^2)$, where $|Y_k|$ represents the number of samples in the k-th site, and r_k denotes the number of clusters obtained after performing TC at the k-th site. However, when employing certain nearest neighbor approximation algorithms, such as the k-d tree, the overall time complexity can be further reduced to $\max_k O(|Y_k| \log |Y_k|) + O(\sum_{k=1}^{S} r_k \log \sum_{k=1}^{S} r_k)$.

III. Experiments

A. Dataset Description

We utilize a total of five synthetic datasets and five real-world datasets for our experiments. The synthetic datasets comprise TB100K [17], CC100K [17], Birch2 [18], Worms-64d [19] and Ring [20]. The real-world datasets include MNIST70K,[1] NASA Shuttle,[2] Notting-Hill [21], Pendigits[3] and Satimage.[4] All these datasets are categorized as either large-scale or medium-scale. For the MNIST70K dataset, we utilize the strong embeddings provided by [22]. Since the distributed clustering methods require subdata provided by each site as input, we partition all the aforementioned datasets into several subdatasets using a split point of 2.5 K, following the common practice. Please refer to Table I for the full statistics of each dataset.

[1][Online]. Available: http://yann.lecun.com/exdb/mnist/

[2][Online]. Available: https://archive.ics.uci.edu/ml/datasets/Statlog+(Shuttle)

[3][Online]. Available: https://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/multiclass.html#pendigits

[4][Online]. Available: https://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/multiclass.html#satimage

TABLE I
FULL STATISTICS OF THE DATASETS UTILIZED

Datasets	Samples	Dimensions	Clusters
TB100K	100k	2	2
CC100K	100k	2	3
Ring	5k	3	2
Birch2	100k	2	100
Worms-64d	105k	64	25
MNIST70K	70k	784	10
Shuttle	58k	9	7
Notting-Hill	4.7k	2000	5
Pendigits	7.5k	16	10
Satimage	4.4k	36	6

B. Experimental Configuration

In the experiments, we conduct a comprehensive comparison of DTC with three mainstream distributed clustering methods, including distributed K-means (DK-means) [1], distributed K-medoids (DK-medoids) [6] and distributed fuzzy C-means (DFCM) [3]. For the other types of distributed clustering methods, such as the distributed density-based clustering methods [4], we do not include them in the comparison due to an absence of open-source codes provided by the authors. However, to provide a more objective assessment of the feasibility or superiority of DTC, we also compare it with three other popular or latest automatic centralized clustering methods, namely, DBSCAN [5], the first integer neighbor clustering hierarchy (FINCH) algorithm [10] and the automatic density peak (DPA) clustering algorithm [13]. For the six algorithms involved in the comparison, most of them include some parameters or hyperparameters that require manual tuning. For DK-means, DK-medoids and DFCM, we set the target number of clusters as the ground truth to maximize their performance advantage. Additionally, due to the stochastic nature of their results, we average the results from 10 runs to obtain the reported results. DBSCAN includes two hyperparameters, namely, Minpts and Eps. We test $Minpts = 10,\ 20, \ldots,\ 50$ and then chose the best setting for each dataset. The Eps settings for all the datasets were determined by the method in [23], i.e., $\bar{d} = \frac{1}{n}\sum_{i=1}^{n} d(x_i, \bar{x})$, where $\bar{x} = \sum_{j=1}^{n} \frac{x_j}{n}$, $d(,)$ denotes the distance, and x_i or x_j denotes a sample. We test $Eps = \bar{d}, \frac{\bar{d}}{2}, \frac{\bar{d}}{3}, \ldots, \frac{\bar{d}}{10}$ and choose the best setting for each dataset. FINCH and DPA have a mechanism for adaptive parameter tuning. We use all the configurations recommended by the authors for these two algorithms. However, DTC does not require any parameter or hyperparameter tuning. It only needs to be applied once to obtain the final results due to its stability. On the other hand, to quantitatively evaluate the results of each algorithm, we employ two widely used external evaluation indices, namely, adjusted normalized mutual information (AMI) [24] and normalized mutual information (NMI) [25]. All experiments were conducted on a workstation with two 32-core AMD EPYC 7532 CPUs (2.4GHz, max 3.33GHz) and 512GB of 3200MHz DDR4-RAM. Our codes are available at https://github.com/brucejak/Distributed-TC.

C. The Results and Analysis

Tables II–III present the quantitative comparison results between DTC and the other six distributed clustering methods or automatic centralized clustering methods, evaluated using AMI and NMI. The best and second-best performances on each dataset are marked in bold and underlined, respectively. As we can see, DTC outperforms previous algorithms on 9 out of 10 datasets. Specifically, in terms of the AMI measurement, DTC exceeds the second-best algorithm by 45.8%, 50.5%, 23%, 12.8% and 12.6% on the TB100K, CC100K, Worms-64d, Shuttle and Notting-Hill datasets, respectively. With respect to the NMI measurement, DTC surpasses the second-best algorithm by 45.8%, 50.5%, 23%, 12.8% and 9.8% on the TB100K, CC100K, Worms-64d, Shuttle and Notting-Hill datasets, respectively.

TABLE II
COMPARISON OF THE CLUSTERING PERFORMANCE OF DTC WITH SIX OTHER DISTRIBUTED CLUSTERING METHODS OR AUTOMATIC CENTRALIZED CLUSTERING METHODS, MEASURED BY AMI

Datasets	DK-means	DK-medoids	DFCM	FINCH	DBSCAN	DPA	DTC
TB100K	.2615	.2898	.2558	.4717	.0004	.5105	**.9681**
CC100K	.0000	.0000	.0000	.4557	.0000	.4941	**.9987**
Ring	.0002	.0040	.0001	.1471	**1.0000**	.5510	**1.0000**
Birch2	.9789	**.9998**	.9290	.9646	.0000	.9940	.9991
Worms-64d	.0275	.4268	.1155	.7158	.3691	.6153	**.9454**
MNIST70K	.8958	.8721	.8889	.4576	.8776	.3806	**.9294**
Shuttle	.2478	.2859	.2531	.0368	.5134	.3700	**.6410**
Notting-Hill	.5940	.6329	.4103	.7545	.6621	.3990	**.8806**
Pendigits	.6639	.6766	.5813	.7538	.4770	.6663	**.7597**
Satimage	.5821	.6327	.6056	.4511	.3527	.5123	**.6422**
Rank	*4.8*	*3.6*	*5.2*	*4.0*	*4.7*	*3.9*	***1.1***

TABLE III
COMPARISON OF THE CLUSTERING PERFORMANCE OF DTC WITH SIX OTHER DISTRIBUTED CLUSTERING METHODS OR AUTOMATIC CENTRALIZED CLUSTERING METHODS, MEASURED BY NMI

Datasets	DK-means	DK-medoids	DFCM	FINCH	DBSCAN	DPA	DTC
TB100K	.2615	.2898	.2558	.4717	.0004	.5105	**.9681**
CC100K	.0000	.0000	.0000	.4557	.0000	.4941	**.9987**
Ring	.0002	.0040	.0001	.3863	**1.0000**	.5510	**1.0000**
Birch2	.9849	**.9999**	.9429	.9817	.0000	.9968	.9991
Worms-64d	.0275	.4793	.1155	.7161	.3691	.6657	**.9459**
MNIST70K	.8958	.8721	.8889	.6560	.8776	.6020	**.9294**
Shuttle	.2478	.2859	.2531	.0368	.5134	.3700	**.6410**
Notting-Hill	.5997	.6336	.4103	.7908	.7871	.6271	**.8884**
Pendigits	.6811	.6855	.6170	.7801	.4770	.7813	**.7863**
Satimage	.5991	.6344	.6123	.5244	.4016	.6070	**.6531**
Rank	*5.0*	*3.7*	*5.3*	*4.1*	*4.7*	*3.4*	***1.1***

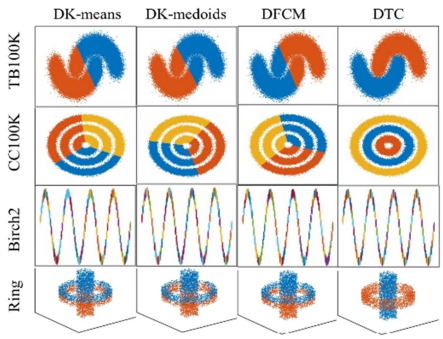

Fig. 3. Visual comparison between DTC and other distributed clustering methods.

From a theoretical standpoint, the less-than-optimal performance of DK-means, DK-medoids and DFCM is fundamentally due to their nature as center (or centroid)-based clustering methods, which are not well suited for nonconvex datasets. The visual comparison in Fig. 3 also demonstrates that DTC possesses the capability to identify clusters with distinct shapes, while the other three distributed clustering algorithms falter in this regard. Concerning the other three automatic centralized algorithms, DBSCAN displays sensitivity to outliers and slight overlaps between clusters, which results in a poor performance on certain large-scale datasets. DPA, which considers density peaks as clustering centers, is unable to accurately recognize clusters exhibiting varying densities. For FINCH, being a hierarchical clustering algorithm rooted in the nearest neighbor chain, its utilization of center linkage to gauge distances between clusters consequently inhibits its effectiveness in identifying nonconvex clusters.

On the other hand, we also employ the average rank to assess the performance of each algorithm across all the datasets. Although most other algorithms require manual tuning of certain parameters

TABLE IV
COMPARISON OF THE CLUSTER NUMBER ESTIMATION CAPABILITY OF DTC WITH THREE OTHER AUTOMATIC CLUSTERING ALGORITHMS, WHERE #C REPRESENTS THE GROUND-TRUTH CLUSTER NUMBER FOR EACH DATASET

Methods/Datasets	TB100K	CC100K	Ring	Birch2	Worms-64d	MNIST70K	Shuttle	Notting-Hill	Pendigits	Satimage
#C	**2**	**3**	**2**	**100**	**25**	**10**	**7**	**5**	**10**	**6**
FINCH	22	125	116	141	**25**	143	16	7	20	5
DBSCAN	1	1	**2**	1	**25**	8	3	21	4	4
DPA	14	60	11	105	110	441	336	63	43	40
DTC	**2**	**3**	**2**	**100**	**25**	**10**	4	**5**	**10**	**6**

TABLE V
INVESTIGATION OF THE IMPACT OF WTC IN THE DTC FRAMEWORK

Measures	Methods/datasets	TB100K	CC100K	Ring	Birch2	Worms-64d	MNIST70K	Shuttle	Notting-Hill	Pendigits	Satimage
AMI	DTC without WTC	.9681	.9987	1.0000	.9991	.9454	.9294	.6371	.8328	.7597	.4300
	DTC	.9681	.9987	1.0000	.9991	.9454	.9294	**.6410**	**.8806**	.7597	**.6422**
NMI	DTC without WTC	.9681	.9987	1.0000	.9991	.9459	.9294	.6371	.8565	.7863	.4300
	DTC	.9681	.9987	1.0000	.9991	.9459	.9294	**.6410**	**.8884**	.7863	**.6531**

or hyperparameters, DTC does not. Nevertheless, it is clear that the average rank of these algorithms is at least three multiplicative factors worse than that of DTC across all the datasets.

D. Automatic Determination of Cluster Numbers

Compared to the other three distributed clustering algorithms, namely, DK-means, DK-medoids and DFCM, DTC can automatically determine the number of clusters. Although there have been some prior density-based distributed clustering algorithms that also possess this capability [4], they accomplish it by introducing new hyperparameters that necessitate manual tuning. Consequently, at its core, adjusting these hyperparameters can be viewed as a form of managing the number of clusters, which invariably leads to considerable inconvenience. However, DTC does not require any hyperparameter tuning and can still automatically determine the number of clusters. To further test DTC's proficiency in estimating ground-truth cluster numbers, we compare it with the three popular or latest centralized automatic clustering methods. As illustrated in Table IV, DTC can estimate the correct number of clusters on nine out of ten datasets, while the other three algorithms accurately predict on only a few datasets. This is because DTC's backbone is torque clustering (TC), and previous research has substantiated TC's potent ability to estimate the number of clusters [16].

E. Impact of Weighted Torque Clustering (WTC)

Within the DTC framework, as illustrated in Fig. 2, we adopt WTC to perform the global clustering step (i.e., Step C) on all the representatives collected from the various sites instead of utilizing TC. This is because following the data reduction step (i.e., Step B), the distribution of the representatives might become sparse. This could potentially impede TC's accuracy in predicting the ground-truth number of clusters, thereby potentially compromising the accuracy of global clustering. In this section, we replace WTC with TC in the DTC framework and observe the variation in the final clustering performance, thereby demonstrating the necessity of WTC's existence. As shown in Table V, the DTC framework without WTC exhibits a performance degradation compared to the original DTC framework on three out of the five real-world datasets, while its performance remains unchanged on five synthetic datasets. This is due to the relatively dense sample distribution of the synthetic datasets; even after a data reduction, the distribution of the representatives remains relatively dense and can be effectively handled by TC. However, the original distribution of the real-world datasets is already relatively sparse. After data reduction, the distribution of the representatives becomes even sparser, thereby affecting the performance of TC in the global clustering stage. Through this ablation study, we observe that the WTC shows more robustness to the sparsity of real-world datasets.

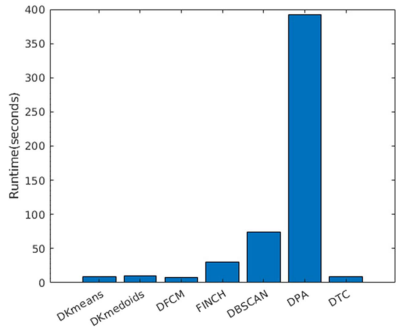

Fig. 4. Runtime comparison between DTC and six state-of-the-art distributed clustering algorithms and automatic centralized clustering techniques on worms-64d dataset.

F. Runtime Comparison

A key advantage of distributed clustering over centralized clustering is its ability to process data in parallel, thereby boosting clustering efficiency. In this section, we compare the running time of DTC with six other algorithms. Typically, the number of samples, dimensions and the number of clusters in a dataset all impact the runtime of the clustering algorithms. Consequently, we choose the Worms-64 dataset, which possesses relatively large values in all three of these aspects, as the test dataset. As shown in Fig. 4, overall, the running times of the four distributed clustering algorithms, namely, DK-means, DK-medoids, DFCM and DTC, are significantly lower than those of the other three centralized algorithms. The running time of DTC is comparable to that of the other three distributed clustering algorithms. However, the clustering accuracy of DTC is significantly superior to that of the other six algorithms, as shown in Tables II–III.

G. Robustness Testing

To comprehensively assess the robustness of the proposed DTC in imbalanced datasets, we followed the methodology of previous work [11] and applied sampling techniques to a representative dataset, Worms-64d. This approach enabled us to generate five imbalanced

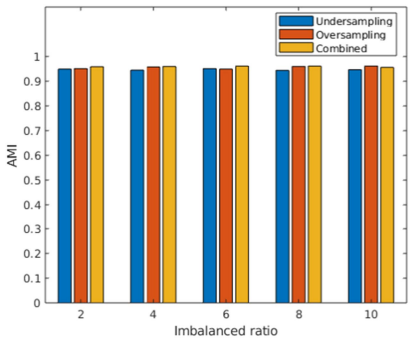

Fig. 5. Performance of DTC on datasets with varying imbalanced ratios.

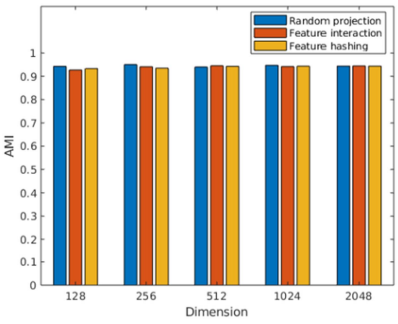

Fig. 6. Performance of DTC on datasets with varying data dimensions.

versions of the Worms-64d dataset with imbalance ratios of 2, 4, 6, 8, and 10. To mitigate the potential bias introduced by the sampling methods, we employed three distinct approaches: Undersampling [26], Oversampling [27], and a combined method of both Undersampling and Oversampling [28]. We continue to utilize the AMI index to quantify the accuracy of the proposed DTC on these imbalanced datasets. As illustrated in Fig. 5, it is evident that the DTC model maintains a consistently high accuracy across the five datasets with progressively increasing imbalance ratios, demonstrating its insensitivity to variations in dataset imbalance. Furthermore, the characteristics of the imbalanced datasets generated by different sampling methods vary. However, this variation does not significantly impact the performance of the DTC model.

To thoroughly evaluate the resilience of the proposed DTC model in the context of high-dimensional datasets, we employed feature engineering techniques on the Worms-64d dataset. This method allowed us to create five enhanced versions of the Worms-64d dataset, each with increasing dimensions of 128, 256, 512, 1024, and 2048. To minimize the biases that might arise from these feature engineering techniques, we adopted three distinct strategies: random projection [29], feature interaction [30], and feature hashing [31]. As depicted in Fig. 6, the DTC model consistently exhibits high accuracy across these enhanced datasets, showcasing its robustness against the challenges posed by high-dimensional data. Despite the varying characteristics of the datasets produced through different feature engineering methods, these differences do not markedly affect the DTC model's performance.

H. Application of Cell Type Identification on Scrna-Seq Datasets

Analyzing clusters in single-cell gene expression profiles aids in uncovering various cell types and the heterogeneity among individual cells [22]. Given that single-cell RNA sequencing (scRNA-seq) data are sensitive in nature, an ideal clustering algorithm for their analysis should possess features that safeguard privacy. In this section, we apply the DTC to clustering tasks on five scRNA-seq datasets [22], comparing it with three aforementioned distributed clustering algorithms, namely DK-means, DK-medoids, and DFCM, as well as a recent centralized, parameter-free clustering algorithm, FINCH. The AMI index is employed to evaluate the accuracy of each algorithm. From Table VI, it is evident that, across the five scRNA-seq datasets, the proposed DTC consistently maintains a performance advantage. Table VII reveals that the DTC is also capable of accurately predicting ground-truth cluster numbers in these scRNA-seq datasets.

TABLE VI
COMPARISON BETWEEN DTC AND OTHER ALGORITHMS ON FIVE SCRNA-SEQ DATASETS, MEASURED BY AMI

Datasets/Methods	DK-means	DK-medoids	DFCM	FINCH	DTC
ALM	.4990	.4631	.4467	.2027	**.6358**
AMB	.4876	.5112	.5118	.1710	**.7082**
NdpKO_R1	.6890	.6601	.6408	.3351	**.7461**
SCINA	.8988	.9004	.8796	.2101	**.9155**
WT_R2	**.7701**	.7063	.5935	.3271	.7682

TABLE VII
COMPARISON OF THE CLUSTER NUMBER ESTIMATION CAPABILITY BETWEEN DTC AND ANOTHER LATEST AUTOMATIC CLUSTERING ALGORITHM ON FIVE SCRNA-SEQ DATASETS, WHERE #C REPRESENTS THE GROUND-TRUTH CLUSTER NUMBER FOR EACH DATASET

Datasets	#C	FINCH	DTC
ALM	7	137	**7**
AMB	4	70	**4**
NdpKO_R1	13	83	10
SCINA	3	110	**3**
WT_R2	13	73	11

On one hand, from the perspectives of accuracy and scalability, the proposed DTC outperforms three distributed clustering algorithms and a recent centralized, parameter-free clustering algorithm on these five real-world scRNA-seq datasets, demonstrating competitive performance advantages. Additionally, the integration of Weighted Torque Clustering (WTC) allows the DTC to determine the number of clusters more accurately in scRNA-seq datasets, particularly considering their high-dimensional sparsity. However, this high-dimensional sparsity significantly impacts another centralized, parameter-free clustering algorithm, i.e., FINCH, leading to over-segmentation in its clustering results, that is, identifying an excessively large number of clusters (see Table VII).

On the other hand, from the perspectives of privacy preservation and data security, the proposed DTC exhibits several advantages over centralized clustering algorithms, such as FINCH and [32]. Firstly, during the data reduction phase of DTC, complete datasets are typically partitioned into multiple segments, each processed on different sites. This means that no single site has access to the entire dataset, thereby reducing the risk of data leakage. Secondly, in the global clustering phase of DTC, the only data exchanged between the hub and various sites are the representative samples from clusters (mean vectors of the clusters) and their weights obtained after clustering at the sites, not the raw data. Thirdly, each site or hub processes only its portion of the data or processed data statistical information. There is no mechanism within DTC that requires full access to all the data, thus reducing the

risk of concentrating large volumes of sensitive data at a single, more vulnerable location.

IV. Conclusion

This paper presents a novel parameter-free distributed clustering framework, namely, distributed torque clustering (DTC). For data or subdata distributed across various sites, DTC primarily carries out two steps. First, DTC applies torque clustering to parallelly cluster the data at each site, extract representatives from each site and subsequently forward all these representatives to a central hub. Second, DTC utilizes weighted torque clustering in the hub to perform global clustering on the representatives, leading to the final partition. We benchmarked DTC against six state-of-the-art distributed clustering algorithms or automatic centralized clustering techniques across ten large-scale or medium-scale datasets, which demonstrated DTC's efficacy and practicability.

References

[1] P. A. Forero, A. Cano, and G. B. Giannakis, "Distributed clustering using wireless sensor networks," *IEEE J. Sel. Topics Signal Process.*, vol. 5, no. 4, pp. 707–724, Aug. 2011, doi: 10.1109/JSTSP.2011.2114324.

[2] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probability*, 1967, pp. 281–297.

[3] Y. Shi, L. Zhang, Z. Cao, M. Tanveer, and C.-T. Lin, "Distributed semisupervised fuzzy regression with interpolation consistency regularization," *IEEE Trans. Fuzzy Syst.*, vol. 30, no. 8, pp. 3125–3137, Aug. 2022, doi: 10.1109/TFUZZ.2021.3104339.

[4] E. Januzaj, H.-P. Kriegel, and M. Pfeifle, "DBDC: Density based distributed clustering," in *Advances in Database Technology-EDBT 2004* (Lecture Notes in Computer Science), E. Bertino, Eds. Berlin, Germany: Springer, 2004, pp. 88–105, doi: 10.1007/978-3-540-24741-8_7.

[5] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowl. Discovery Data Mining*, Aug. 1996, pp. 226–231.

[6] M.-F. F. Balcan, S. Ehrlich, and Y. Liang, "Distributed k-means and k-median clustering on general topologies," in *Proc. 26th Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 1995–2003.

[7] M. Bendechache and M.-T. Kechadi, "Distributed clustering algorithm for spatial data mining," in *Proc. IEEE 2nd Int. Conf. Spatial Data Mining Geographical Knowl. Serv.*, 2015, pp. 60–65, doi: 10.1109/ICSDM.2015.7298026.

[8] M. Klusch, S. Lodi, and G. Moro, "Distributed clustering based on sampling local density estimates," in *Proc. 18th Int. Joint Conf. Artif. Intell.*, 2003, pp. 485–490.

[9] D. K. Tasoulis and M. N. Vrahatis, "Unsupervised distributed clustering," in *Proc. IASTED Int. Conf. Parallel Distrib. Comput. Netw.*, 2004, pp. 347–351.

[10] S. Sarfraz, V. Sharma, and R. Stiefelhagen, "Efficient parameter-free clustering using first neighbor relations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 8926–8935, doi: 10.1109/CVPR.2019.00914.

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

[12] 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, doi: 10.1109/TPAMI.2019.2924953.

[13] 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, doi: 10.1016/j.ins.2021.01.010.

[14] 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.

[15] J. Hou, H. Yuan, and M. Pelillo, "Towards parameter-free clustering for real-world data," *Pattern Recognit.*, vol. 134, Feb. 2023, Art. no. 109062, doi: 10.1016/j.patcog.2022.109062.

[16] J. Yang and C.-T. Lin, "Autonomous clustering by fast find of mass and distance peaks," May 2022, doi: 10.36227/techrxiv.19691914.v1.

[17] T. Qiu and Y.-J. Li, "Fast LDP-MST: An efficient density-peak-based clustering method for large-size datasets," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 5, pp. 4767–4780, May 2023, doi: 10.1109/TKDE.2022.3150403.

[18] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: A new data clustering algorithm and its applications," *Data Mining Knowl. Discov.*, vol. 1, no. 2, pp. 141–182, Jun. 1997, doi: 10.1023/A:1009783824328.

[19] S. Sieranoja and P. Fränti, "Fast and general density peaks clustering," *Pattern Recognit. Lett.*, vol. 128, pp. 551–558, Dec. 2019, doi: 10.1016/j.patrec.2019.10.019.

[20] A. Lotfi, P. Moradi, and H. Beigy, "Density peaks clustering based on density backbone and fuzzy neighborhood," *Pattern Recognit.*, vol. 107, Nov. 2020, Art. no. 107449, doi: 10.1016/j.patcog.2020.107449.

[21] B. Wu, Y. Zhang, B.-G. Hu, and Q. Ji, "Constrained clustering and its application to face clustering in videos," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 3507–3514, doi: 10.1109/CVPR.2013.450.

[22] D. Peng et al., "Clustering by measuring local direction centrality for data with heterogeneous density and weak connectivity," *Nature Commun.*, vol. 13, no. 1, Sep. 2022, Art. no. 5455, doi: 10.1038/s41467-022-33136-9.

[23] L. Bai, X. Cheng, J. Liang, H. Shen, and Y. Guo, "Fast density clustering strategies based on the k-means algorithm," *Pattern Recognit.*, vol. 71, pp. 375–386, Nov. 2017, doi: 10.1016/j.patcog.2017.06.023.

[24] 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.

[25] A. Strehl and J. Ghosh, "Cluster ensembles–a knowledge reuse framework for combining multiple partitions," *J. Mach. Learn. Res.*, vol. 3, pp. 583–617, 2002.

[26] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory undersampling for class-imbalance learning," *IEEE Trans. Syst., Man, Cybern., Part B Cybern.*, vol. 39, no. 2, pp. 539–550, Apr. 2009, doi: 10.1109/TSMCB.2008.2007853.

[27] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002, doi: 10.1613/jair.953.

[28] N. Junsomboon and T. Phienthrakul, "Combining over-sampling and under-sampling techniques for imbalance dataset," in *Proc. 9th Int. Conf. Mach. Learn. Comput.*, 2017, pp. 243–247, doi: 10.1145/3055635.3056643.

[29] E. Bingham and H. Mannila, "Random projection in dimensionality reduction: Applications to image and text data," in *Proc. 7th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2001, pp. 245–250, doi: 10.1145/502512.502546.

[30] R. Jain and W. Xu, "HDSI: High dimensional selection with interactions algorithm on feature selection and testing," *PLoS One*, vol. 16, no. 2, Feb. 2021, Art. no. e0246159, doi: 10.1371/journal.pone.0246159.

[31] C. B. Freksen, L. Kamma, and K. Green Larsen, "Fully understanding the hashing trick," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 5394–5404.

[32] H. Liu et al., "Robust collaborative clustering of subjects and radiomic features for cancer prognosis," *IEEE Trans. Biomed. Eng.*, vol. 67, no. 10, pp. 2735–2744, Oct. 2020, doi: 10.1109/TBME.2020.2969839.

[33] Z. Shi, L. Chen, W. Ding, C. Zhang, and Y. Wang, "Parameter-free robust ensemble framework of fuzzy clustering," *IEEE Trans. Fuzzy Syst.*, vol. 31, no. 12, pp. 4205–4219, Dec. 2023, doi: 10.1109/TFUZZ.2023.3277692.

[34] D. Wu, F. Nie, X. Dong, R. Wang, and X. Li, "Parameter-free consensus embedding learning for multiview graph-based clustering," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 12, pp. 7944–7950, Dec. 2022, doi: 10.1109/TNNLS.2021.3087162.

[35] X. Li, H. Zhang, R. Wang, and F. Nie, "Multiview clustering: A scalable and parameter-free bipartite graph fusion method," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 1, pp. 330–344, Jan. 2022, doi: 10.1109/TPAMI.2020.3011148.

[36] C. Wu, X. Zhang, and S. Yan, "Parameter-free auto-weighted possibilistic fuzzy C-means clustering with kernel metric and robust algorithm," *SN Comput. Sci.*, vol. 4, no. 4, May 2023, Art. no. 411, doi: 10.1007/s42979-023-01824-y.