# SBSC: A fast Self-tuned Bipartite proximity graph-based Spectral Clustering

ABDUL ATIF KHAN, PDPM Indian Institute of Information Technology, Design and Manufacturing, Jabalpur, India
RASHMI MAHESHWARI, PDPM Indian Institute of Information Technology, Design and Manufacturing, Jabalpur, India
MOHAMMAD MAKSOOD AKHTER, PDPM Indian Institute of Information Technology, Design and Manufacturing, Jabalpur, India
SRABAN KUMAR MOHANTY, PDPM Indian Institute of Information Technology, Design and Manufacturing, Jabalpur, India

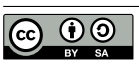
Spectral clustering (SC) is well-known for discovering natural groups present in the data by projecting them into Eigen-space based on the proximity graph but incurs cubic time in terms of size ($N$) of the data as all pair proximity is used. To enhance the efficiency of the SC techniques, the proximity between the data instances and their representatives ($R$) is captured through a bipartite similarity graph. However, extrinsic parameters such as the number of representatives and nearby representatives of data instances, influence the clustering performance, time, and memory usage. Therefore, in this work, we construct a parameter-free bipartite graph to further improve the clustering quality and computational cost of SC by introducing a locality-based sparsification technique. First, the proposed method (SBSC) determines $O(\sqrt{N})$ numbers of well-distributed representatives in $O(N \lg N)$ time by applying Bi-means and $K$-means partitioning techniques. Next, SBSC utilizes the local neighbors of $R$ to search the nearby representatives, which fastens the search time to $O(N)$. To the best of our knowledge, the proposed bipartite graph is the least ($O(N)$) sized and therefore, by exploiting the high sparsity of the graph accelerates the Eigen-decomposition step of SBSC. The proposed algorithm takes overall $O(N(K^2 + \lg N))$ time only to detect $K$ clusters, and experimental results on eighteen large-sized diversified datasets suggest that SBSC discovers complex clusters much faster than the competing methods with enhanced clustering quality.

CCS Concepts: • **Information systems → Clustering**; **Nearest-neighbor search**; • **Theory of computation → Unsupervised learning and clustering**; **Graph algorithms analysis**; • **Computing methodologies → Cluster analysis**; **Spectral methods**.

Additional Key Words and Phrases: Bipartite proximity graph, Parameter-free, Large-scale spectral clustering, Eigenvectors projection, Approximate searching

**ACM Reference Format:**
Abdul Atif Khan, Rashmi Maheshwari, Mohammad Maksood Akhter, and Sraban Kumar Mohanty. 2025. SBSC: A fast Self-tuned Bipartite proximity graph-based Spectral Clustering. *Proc. ACM Manag. Data* 3, 3 (SIGMOD), Article 213 (June 2025), 22 pages. https://doi.org/10.1145/3725418

Authors' Contact Information: Abdul Atif Khan, Computer Science and Engineering, PDPM Indian Institute of Information Technology, Design and Manufacturing, Jabalpur, Madhya Pradesh, India, k.atif@iiitdmj.ac.in; Rashmi Maheshwari, Computer Science and Engineering, PDPM Indian Institute of Information Technology, Design and Manufacturing, Jabalpur, Madhya Pradesh, India, m.rashmi@iiitdmj.ac.in; Mohammad Maksood Akhter, Computer Science and Engineering, PDPM Indian Institute of Information Technology, Design and Manufacturing, Jabalpur, Madhya Pradesh, India, maksood@iiitdmj.ac.in; Sraban Kumar Mohanty, Computer Science and Engineering, PDPM Indian Institute of Information Technology, Design and Manufacturing, Jabalpur, Madhya Pradesh, India, sraban@iiitdmj.ac.in.




ACM 2836-6573/2025/6-ART213
https://doi.org/10.1145/3725418

## 1 Introduction

Clustering a high number of data instances is becoming popular in this digital era due to several real-life applications. A clustering task aims to identify groups of similar data points in the absence of any ground truth labels. However, clustering large-scale data (like grouping long texts [17, 35], clustering images [25, 36], image segmentation [1, 26], sparse data clustering [14], etc.) with limited resources is a challenging task. In the past few years, several spectral clustering (SC) methods have shown promising results in this area [2, 11, 13, 42, 46].

The first step of conventional SC constructs a sparse proximity graph by treating the data points as nodes and edges representing the dis(similarity) between them [30, 39, 41]. For example, connecting each data point to its $k$-nearest neighbors is a popular technique to build the proximity graph [41]. The next step computes the Laplacian matrix ($L$) by exploiting the locality information of the nodes using the degree matrix and affinity between nodes. Here, $L$ is a symmetric matrix, which means the eigenvalues of $L$ are real and therefore, are comparable. Then $K$-eigenvectors corresponding to the $K$-smallest eigenvalues of $L$ are computed to transform the data into a $K$-dimensional space ($U$). Lastly, the $K$-means [16] is applied on $U$ to get the final clusters. However, the conventional SC initially considers all-pairs distances between the data points to construct the sparse graph which takes quadratic time and memory, and further constructing $U$ requires $O(N^3)$ time, where $N$ is the number of data points.

A popular approach to make SC feasible for large-scale problems is considering the proximity between the data points and $p$ number of representatives, s.t., the $p \ll N$ [2, 13, 26]. The structural similarity of the data is captured by a bipartite proximity graph $B = \{V, E, W\}$. Here, $V$ is the the set of nodes defined as $V = X \cup R$, s.t., $X \cap R = \phi$, where $X$ and $R$ are the set of data points and representatives, respectively. $E$ is the set of edges that connect each data point to its nearby representatives, and $W \in \mathbb{R}^{N \times p}$ is the affinity sub-matrix that defines the weight of each edge. $W$ plays a vital role in a bipartite graph-based spectral clustering (BSC) method to overcome the computational overhead of the Eigen-decomposition step. The efficient techniques like the Transfer-cuts [26] solves the Eigen-problem of BSC faster by generating the smaller affinity matrix ($W_R \in \mathbb{R}^{p \times p}$) for representative points from $W$. The graph Laplacian ($L_R$) of $W_R$ is a symmetric real-valued matrix. The $K$-eigenvectors corresponding to the $K$-smallest eigenvalues of $L_R$ suggests the approximation of the corresponding eigenvalues and eigenvectors of the Laplacian matrix of graph $B$, resulting in the efficient construction of the transformed space $U$.

A straightforward way to determine the representative points is by randomly selecting the $p$ data points as representatives [2]. For better clustering quality, another strategy divides the data into $p$ sub-clusters using the $K$-means to treat their centers as representatives but takes $O(Np)$ execution time [2]. Also, like the conventional similarity graphs, the bipartite graph needs sparsification, for which $k$ nearest representatives for every data point are determined and are connected via edges [2, 13]. However, the size of the bipartite graph becomes $O(Nk)$ and construction requires $O(Np)$ time due to searching over $p$ representatives for every data point [2].

Recently, several BSC methods were proposed to further accelerate the construction time of the bipartite graph by introducing faster sub-clustering strategies and locating the approximate representatives [13, 23]. Table 1 shows the stage-wise time complexity and size of the bipartite graph of such existing methods. U-SPEC [13] is a popular technique that uses both the random selection and $K$-means to compute $p$ representatives in only $O(p^2)$ time. DnC [23] is a recent method that searches the $k$-nearest representatives of a data point only within the $(k \times 10)$ number of local representatives, costing $O(Nk)$ time. However, the clustering quality and computational complexities of such existing BSC methods depend upon the extrinsic parameters like the number of representative points ($p$) and nearest representatives ($k$). Moreover, there is also a randomness

in their methods of finding $R$. Due to the tuning of parameters and the randomness, most BSC methods require executing the algorithms multiple times to get the desired results, which is a time-consuming process when dealing with large-sized datasets [2, 13, 23].

This work proposes a fast Self-tuned Bipartite proximity graph-based Spectral Clustering (SBSC) to overcome the discussed bottlenecks of the large-scale SC. Firstly, SBSC determines $O(\sqrt{N})$ numbers of structurally preserved representatives using a fast hierarchical divisive-based partitioning technique (Bi-means [19]) and the $K$-means. Then, the next step computes the two-hops neighbors of the representatives using two rounds of the minimum spanning tree (MST) on $R$ to quickly yet effectively locate the nearby representatives of data instances for constructing the linear-sized bipartite graph. After the construction of the graph, SBSC applies the Transfer-cuts algorithm along with the Lancsoz method [10] to obtain the transformed space $U$ efficiently, and at last, executes the $K$-means algorithm on $U$ to discover the final $K$ clusters. Below are the primary contributions of SBSC.

(1) We propose a bipartite proximity graph that intrinsically captures the topological information of data objects using the MST-based locality of well-scattered representatives to enhance the clustering quality of SC.
(2) The proposed technique of determining the sub-linear representatives and the MST-based proximity computation makes the similarity graph deterministic and independent of any extrinsic parameters, i.e., it is not required to run the algorithm multiple times to tune the similarity graph for the given dataset.
(3) The usage of hierarchical divisive-based recursive partitioning determines the quality of representative points in only linearithmic, i.e., $O(N \lg N)$ time, and the local neighborhood search using the MST of $R$ finds the neighbors of points in only $O(N)$ time which is faster than the existing methods. Furthermore, we also exploit the high sparsity of the proposed graph to accelerate the Transfer-cuts method for computing the Eigen-space.
(4) SBSC takes overall $O(N(K^2 + \lg N))$ time to detect clusters with only $O(N)$ space. To the best of our knowledge, this is the most efficient BSC method in terms of the space complexity, and empirical studies over the eighteen large-sized datasets show that SBSC outperforms the several fast benchmark and state-of-the-art SC techniques in terms of the overall execution time, clustering quality, graph's size, and the number of parameters.

Section 2 discusses methods related to large-scale SC, Section 3 explains the step-by-step procedure of SBSC, Section 4 shows the theoretical analyses of SBSC, Section 5 demonstrates the experimental results, and Section 6 discusses the conclusion and future direction of SBSC.

Table 1. Stage-wise time complexity and the graph size of the competing bipartite graph-based spectral clustering methods along with the external parameters required by them.

| **Methods** | **External parameters** | **Representative points selection time** | **Bipartite graph construction time** | **Eigen-decomposition time** | **Size of the bipartite graph** |
|---|---|---|---|---|---|
| LSC-R [2] | $p, k, K$ | – | $O(Np)$ | $O(Np^2 + p^3)$ | $O(Nk)$ |
| LSC-K [2] | $p, k, K$ | $O(Np)$ | $O(Np)$ | $O(Np^2 + p^3)$ | $O(Nk)$ |
| U-SPEC [13] | $p, k, K$ | $O(p^2)$ | $O(Np^{1/2})$ | $O(Nk(k+K) + p^3)$ | $O(Nk)$ |
| DnC [23] | $p, k, \alpha, K$ | $O(N\alpha)$ | $O(Nk)$ | $O(Nk(k+K) + p^3)$ | $O(Nk)$ |
| SBSC | $K$ | $O(N \lg N)$ | $\boldsymbol{O(N)}$ | $O(KN + K^2N^{1/2})$ | $\boldsymbol{O(N)}$ |

Symbols $N$, $K$, $p$, and $k$ denote the number of data points, clusters, representative points, and neighbors respectively. $\alpha$ is a parameter used in DnC to decide $K$ for $K$-means.
**Note:** The above analyses ignore the number of features ($F$).
The post-clustering time for all of the above methods is $O(NK^2)$.

## 2 Related work

The conventional SC methods identify the complex-shaped clusters by capturing the topological structure of the data using the sparse proximity graph [39, 41]. Effective construction of the similarity graph is a key step of SC to achieve satisfactory results [24, 41]. Ng et al. [30] employed the Gaussian kernel function in their similarity graph to decide the proximity ($W_{ij}$) between any two data points ($x_i$ and $x_j$). The $k$-nearest neighbors ($k$NN) graph is another well-known proximity graph that connects only $k$-closest neighbors of each data instance [41]. However, the conventional SC methods need $O(N^2)$ time & space to construct the graph and cubic time for the Eigen-decomposition step [41].

To speed up SC, parallelism was introduced to SC in several works [3, 15, 38]. Chen et al. [3] proposed a parallel SC (PSC) that splits data objects over distributed machines, and every machine performs the Eigen-decomposition task for lesser data objects. Similarly, Huo et al. [15] speed up SC by efficiently parallelizing the affinity matrix and post-clustering ($K$-means) process.

The time and space requirement of SC can be reduced by utilizing the representative points ($R$) to construct the proximity graph, s.t., the number of representatives ($p$) $<< N$ [2, 45]. One of the popular methods is approximate spectral clustering (ASC) that builds the proximity graph only on $R$ as the nodes to reduce the size of the graph [4, 21, 40, 45]. The SC is applied on the reduced graph to determine the clusters of $R$, further suggesting the clusters of data instances. ASC reduces the graph construction time and space to $O(p^2)$ and the time complexity of the Eigen-decomposition process is reduced to $O(p^3)$ [45].

However, the drawback of ASC is that its clustering results mainly depend upon the quality of representative points. If a representative point ($\mu_j$) is mis-clustered, all the data instances associated with $\mu_j$ will also belong to the wrong clusters. In the past years, another category of SC has emerged that is less sensitive to $R$ by considering the proximity between the data instances and representatives [2, 5, 9, 13]. So, the structural similarity is captured by a bipartite proximity graph $B$ in which the vertex set is divided into two disjoint subsets, i.e., the set of data instances ($X$) & $R$, and edges connect the data points to their nearby representatives whose weights are defined by the affinity sub-matrix $W$ of order $(N \times p)$. The affinity value $W_{il} \in W$ defines the proximity between the data point $x_i \in X$ and the representative $\mu_l \in R$. We term such SC methods utilizing a bipartite graph as the "Bipartite graph-based Spectral Clustering (BSC)".

BSC methods accelerate the Eigen-decomposition step by computing the approximate eigenvectors of the graph Laplacian of $B$ using the eigenvectors of the smaller matrix of order $(p \times p)$, which is derived from the sub-matrix $W$ [2, 13]. Fowlkes et al. [9] performed image segmentation using the pairwise proximity between the pixels and few randomly sampled pixels. They incorporated Nystrom approximation to efficiently project the data into the Eigen-space. Similarly, Li et al. [26] introduced the Transfer-cuts method in BSC to perform the image segmentation task more effectively. Transfer-cuts projects the Eigen-space ($U$) efficiently by utilizing the $K$-eigenvectors corresponding to the $K$-smallest eigenvalues of the Laplacian matrix ($L_R \in \mathbb{R}^{p \times p}$) of the reduced graph having nodes as $R$.

To perform clustering on large-sized datasets, Cai et al. [2] suggested a novel BSC algorithm (LSC) that constructs the bipartite graph by connecting each data point to its $k$-closest landmarks/representatives and then applies the sparse coding-based approach to find the eigenvectors, but it requires $O(Np^2 + p^3)$ time. Recently, Hong et al. [12] proposed a modified version of the LSC method by introducing an additional step of building the low-rank affinity matrix using the probability density estimators.

Huang et al. [13] proposed a BSC method (U-SPEC) that applies $K$-means on the sampled points to determine $R$ faster. Furthermore, U-SPEC tuned the $k$-closest representatives of the instances using

an approximate searching technique that considers the sub-clusters of representatives, which brings down the graph construction time to $O(p^2 + Np^{1/2})$. Similarly, Li et al. [23] determined $R$ using the divide and conquer based sub-grouping approach. Their clustering method (DnC) determined the nearest representatives locally using the neighborhood of sub-clusters that reduces the bipartite graph construction time to $O(Nk)$. Both U-SPEC and DnC follow the Transfer-cuts method to project the Eigen-space because of its high efficiency as well as apt utilization of the neighborhood properties of the bipartite graph. If each data point is connected with $k$-representatives then the Transfer-cuts algorithm takes only $O(Nk(k + K) + p^3)$ time to detect $K$ clusters [13].

### 2.1 Problem formulation

Graph-based clustering techniques [19, 41] aim to divide the given set ($X = \{x_1, x_2, \cdots x_N\} \in \mathbb{R}^{N\times F}$) of $F$-dimensional data objects into $K$-clusters ($C = \{C_1, C_2, \cdots, C_K\}$) by exploiting the structural similarity of the data using the proximity graph $G_S = \{V_S, E_S, A\}$, s.t., objects as the set of vertices ($V_S$), the set $E_S$ consists edges between the adjacent points, and $A \in \mathbb{R}^{N\times N}$ represents the affinity matrix to measure the proximity ($A_{ij}$) between each pair of connected objects ($x_i$ and $x_j$). Minimizing graph cut problems like $NCut$ [39, 41] could detect quality clusters, defined as

$$NCut = \sum_{k=1\ to\ K} \frac{cut(C_k, \overline{C_k})}{size(C_k)} \tag{1}$$

where, $\overline{C_k} = V_S \setminus C_k$

$$cut(C_k, \overline{C_k}) = \sum_{x_i\in C_k,\ x_j\in\overline{C_k}} A_{ij}$$

$$size(C_k) = \sum_{x_i\in C_k} \left( \sum_{x_j\in X} A_{ij} \right)$$

Here, $cut(C_k, \overline{C_k})$ is the sum of weights of inter-edges connecting the data points of the cluster $C_k$ with the data points of the complement ($\overline{C_k}$) of $C_k$, and $size(C_k)$ is the sum of the degree-sum of each node/object present in $C_k$. The degree-sum of a data object ($x_i$) is the sum of edge weights between $x_i$ and its neighbors, i.e., $\sum_{x_j\in X} A_{ij}$.

$NCut$ is an NP-hard problem, so SC came into the picture to relax $NCut$ by considering the generalized eigenvectors ($f$) associated with the first generalized eigenvalues ($\gamma$) of the following equation [41]:

$$Lf = \gamma Df \tag{2}$$

where $D$ is the degree matrix of $G_S$ and $L = D - A$ is the unnormalized graph Laplacian [29, 41]. However, determining $f$ takes cubic computational time [41], i.e., costly for large-sized datasets.

To overcome the computational bottleneck, several SC methods utilize the bipartite proximity graph ($B$), which is constructed on data points and $p$-representatives ($R$) as nodes by connecting each object to its neighboring representatives. Therefore, the affinity between the nodes is now represented by the sub-matrix $W \in \mathbb{R}^{N\times p}$, which is smaller than $A$, and methods like Transfer-cuts [26] can utilize $W$ to solve Eq. (2) efficiently. (Please refer to Subsection 3.3.1 for the detailed steps of Transfer-cuts).

However, the clustering performance and computational complexities of BSC are dependent upon some extrinsic factors like the number of representatives ($p$) and nearby representatives ($k$). Table 1 mentions the number of external parameters used along with the time and space

complexities of some of the existing BSC methods. The table clearly suggests that existing methods are required to tune some parameters to suitably construct the bipartite graph for the discovery of the quality clusters, which sometimes becomes a challenging as well as time-consuming process for large datasets.

In the past, several graph clustering methods suggested that the MST appropriately captures the internal structure of the data points in fewer edges without tuning any parameters [22, 33, 47]. To the best of our knowledge, the usage of MST-based proximity for constructing the bipartite proximity graph is still unexplored. So, this work focuses on removing the discussed parameters of the bipartite neighborhood graph by determining the well-spread out representative points and collecting the nearby edges via MST. Also, the proposed work aims to further improve the time and space complexities of BSC by applying a fast linearithmic time partitioning approach to compute representatives and utilizing the two-hops-based neighborhood of $R$ to locate the neighbors of data instances.

## 3 Proposed Approach

The proposed SC technique (SBSC) aims to suitably group similar data instances by constructing the bipartite similarity graph between the set of data instances ($X$) and representative points ($R$) without considering any extrinsic factors. We achieve this by first computing the sub-linear amount of quality representative points using a fast partitioning strategy, then efficiently finding the nearby representatives of the data objects based on the local information of $R$ using MST. The usage of MST makes the proximity graph independent of the sparsity parameter, and only a linear amount of time & space are required for constructing the graph.

Given a dataset $X \in \mathbb{R}^{N \times F}$, where $N$ and $F$ are the number of data points and features respectively. The steps of SBSC to detect the $K$ clusters of $X$ are given below:

(1) Firstly, $X$ is divided into $O(\sqrt{N})$ number of partitions using the Bi-means and $K$-means techniques, and their centers are considered as the set of representatives ($R$).
(2) The next step finds the neighbors and two-hops neighbors of the representative points by computing two rounds of MST on the complete graph of $R$.
(3) A sparse bipartite graph $B$ is built between $X$ and $R$ by finding the nearest representatives of each data point based on the two-hops neighbors of $R$.
(4) The Transfer-cuts along with the Lancsoz method is applied using the affinity sub-matrix of the bipartite graph to form the Eigen-space $U \in \mathbb{R}^{N \times K}$.
(5) Lastly, $K$-means is applied on $U$ to get the final clusters.

Fig. 1 visually shows the steps of SBSC. To calculate the distance between any two points ($x_i$ and $x_j$), we use the Euclidean distance, denoted as $d(x_i, x_j)$. The following subsections describe the steps of SBSC in more detail.

### 3.1 Determining representative points

The number of representatives ($p$) should be sufficient to capture the shape of clusters well, however, a higher value of $p$ can increase the computational cost. For example, the next phase computes the pairwise distances among $R$ which requires $O(p^2)$ time. Therefore, SBSC selects $p = O(\sqrt{N})$ representatives to ensure that a sufficient number of representatives are captured and the running time of the upcoming phase is also within the limit of $O(N)$. SBSC determines the well-scattered representatives by generating the $O(\sqrt{N})$ number of partitions of the data. Firstly, SBSC applies the Bi-means [19] method to discover the partitions faster. Bi-means follows the top down approach that recursively splits the data into two partitions based on the two border points that are far apart from one another. Initially, Bi-means considers the whole dataset ($X$) as a single partition, then

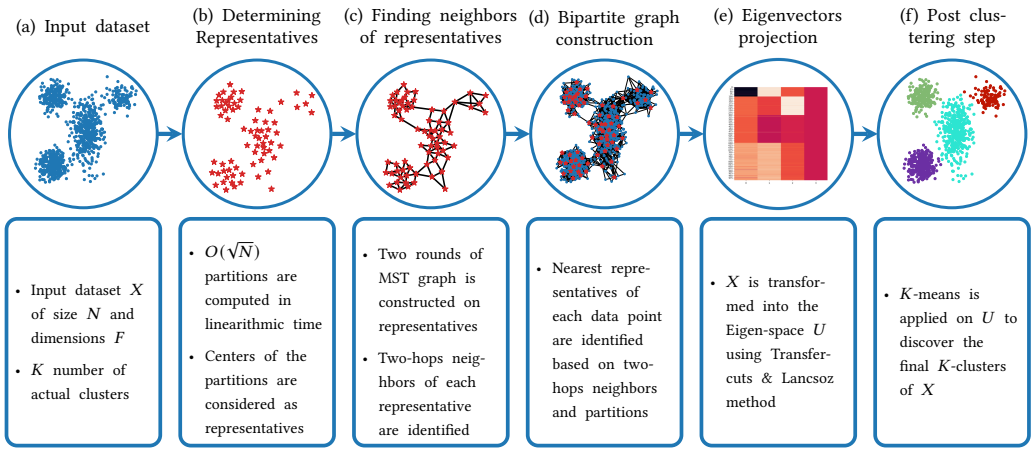



Fig. 1. Phases of the proposed method (SBSC). (a) A sample two-dimensional dataset ($X$) with four clusters. (b) The representative points, computed by SBSC. (c) The edges after applying two rounds of MST. (d) The edges of the proposed bipartite graph. (e) The heatmap of matrix $U$ which shows the variation in the values along four features of the transformed space. (f) The final clusters discovered by SBSC.

finds the two border points, s.t., the first border point ($b_1$) is the point farthest from the center point ($\mu$) of $X$, and the second border point ($b_2$) is the most distant point from $b_1$. Fig. 2a visualizes $b_1$ and $b_2$ of the sample dataset. The next step splits $X$ into two partitions by assigning each data instance to its closest border point and perform the same splitting process recursively till the discovery of the desired number of partitions.

The Bi-means method is efficient and takes only $O(N \lg N)$ time to discover the partitions. However, there is a possibility that some partitions posses data points of different clusters. So, to achieve the quality of partitions faster, we perform partitioning in two stages. The first stage uses the Bi-means method to speed up the partitioning process by recursively splitting the data till the size of each partition becomes less than ($k' \times \sqrt{N}$), i.e., till we get $O(\sqrt{N}/k')$ number of intermediate partitions, where $k'$ is a small positive integer. We empirically find that $k' = 10$ is sufficient to get the quality partitions by maintaining the low computational time.

Fig. 2b shows the intermediate partitions ($P' = \{P'_1, P'_2, \cdots, P'_{p'}\}$) generated after the first stage. The next stage applies the $K$-means algorithm to discover more effective partitions. However, employing the standard $K$-means brings randomness to the proposed technique, i.e., requires running the method several times to get the desired results. We overcome this bottleneck by utilizing the Bi-means algorithm to initialize the cluster centers for $K$-means.

On each intermediate partition ($P'_j$), the Bi-means method is applied again to get $O(k')$ sub-partitions and their centroids ($C'$). Fig. 2c shows the centroids ($C'$) computed by the Bi-means method for a given intermediate partition of the sample dataset. Now, $P'_j$ is divided into $O(k')$ number of sub-partitions using $K$-means by considering $C'$ as the initial center. Fig. 2d visualises the sub-partitions of $P'_1$ discovered by the $K$-means in the second stage.

Stage-2 produces total $O(\sqrt{N})$ partitions. Then the centroid of each partition $P_j$ is considered as the representative of all points that belong to $P_j$. Fig. 2e & 2f show the final $O(\sqrt{N})$ partitions and their representatives, respectively, indicating that the proposed representative points appropriately preserve the shapes of the original clusters. Algorithm 1 describes the steps of SBSC to find the

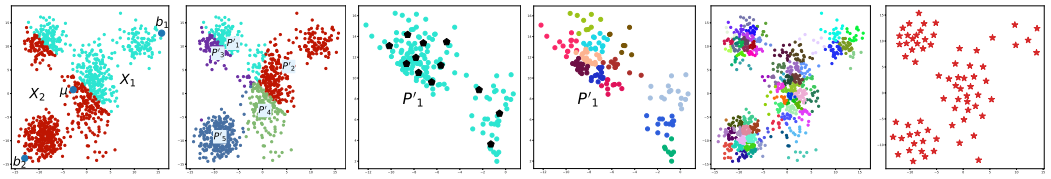



Fig. 2. Determining representatives of the input data ($X$): (a) Splitting of $X$ based on two border points. (b) Intermediate partitions produced by the Bi-means at the first stage. (c) Initial cluster centers ($C'$) for an intermediate partition ($P'_1$) using Bi-means. (d) $O(k')$ number of sub-partitions of $P'_1$ using the $K$-means method by considering $C'$ as the initial centers. (e) Final partitions of $X$ after applying stage-2 partitioning. (f) Final representative points ($R$).

representative points by computing the partitions for the given dataset. It returns two sets, i.e., the set of partitions $P = \{P_1, P_2, \cdots P_p\}$, and representative points $R = \{\mu_1, \mu_2, \cdots \mu_p\}$.

**Algorithm 1:** Determining representatives

**Data:** Dataset $X$ with $N$ number of points.
**Result:** Set of representative points $R$ and the set of partitions $P$.

1 Apply Stage-1 partitioning. // $O(N \lg N)$ time
  Apply the Bi-means method on dataset $X$ to get the $O(\sqrt{N}/k')$ number of intermediate partitions, i.e., $P' = \{P'_1, P'_2, \cdots, P'_{p'}\}$.
2 Apply Stage-2 partitioning. // $O(N \lg N)$ time
  Initialize, $P \leftarrow \phi$ and $R \leftarrow \phi$
  **foreach** *intermediate partition* $P'_j \in P'$ **do**
    Apply the Bi-means method for $P'_j$ to get $O(k')$ sub-partitions and their center points ($C'$).
    By considering $C'$ as the initial centers, apply the $K$-means on $P'_j$, to get the final sub-partitions of $P'_j$.
    Add the sub-partitions that are computed by the previous step to the set $P$.
  Compute the center points of all the partitions present in $P$, and treat them as the set of representative points $R$.
3 **return** *$R$ and $P$.*

### 3.2 Constructing the bipartite proximity graph

After computing $R$, each data instance is connected to its nearby representatives to construct a sparse proximity graph for the SC algorithm. However, this takes $O(Np)$ computational time by the brute force technique. SBSC fastens the process by first determining pairs of neighboring representatives and then performing the local search to detect the nearby representatives of a data instance ($x_i$) based on the neighbors of its representative point, in order to avoid the searching over all the representatives to find the neighbors of $x_i$.

*3.2.1 Neighboring representatives.* We use the MST to locate the neighbors of representative points. MST spans all the $p$ nodes of the graph using $(p-1)$ edges of minimum sum of weights, which

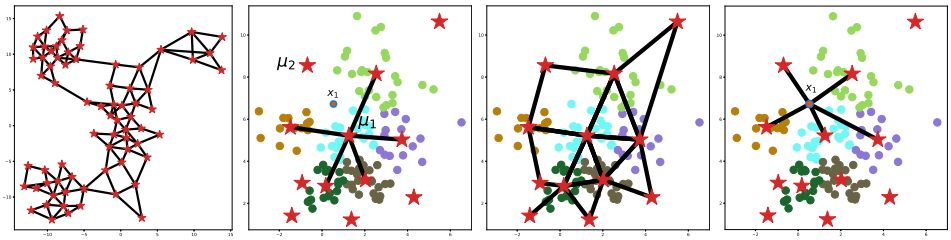



Fig. 3. Locating the nearest representatives of the data instances (a) Two rounds of MST ($M_2$) of the complete graph of $R$. (b) The neighbors ($NN_1$) of a representative point $\mu_1$. (c) The two-hops neighbors ($NN_1^2$) of $\mu_1$. (d) The nearest representatives of a data instance ($x_1$).

allows MST to exploit the topological structure of the data while capturing the neighbors of the nodes.

SBSC computes the MST ($T_1$) of the complete graph whose nodes are represented by the set of representative points $R$, but $T_1$ may not be sufficient enough to exploit the local information of the representatives. So, we also include the edges of the second round of MST ($T_2$), which is the MST of the complete graph of $R$ without including the edges of $T_1$. The edges of $T_1$ and $T_2$ are merged to get the neighborhood graph ($M_2$) of representative points. Fig. 3a illustrates the graph $M_2$ for the given representatives, and we can see that its edges connect the closer representatives. Next, the immediate ($NN_j$) and two-hops ($NN_j^2$) neighbors of every representative point ($\mu_j$) are identified using $M_2$. For a representative $\mu_j$, its neighbors are the representatives connected to it by the edges of $M_2$, and its two-hops neighbors include all the neighbors along with the neighbors of neighbors. Figs. 3b and 3c visualize the neighbors and two-hops neighbors of a representative point $\mu_1$, respectively.

*Definition 3.1 (Neighbors and two-hops neighbors of a representative).* If there is a representative point $\mu_j \in R$, then the set of neighbors ($NN_j$) of $\mu_j$ is all the representatives connected to it through the edges ($E(M_2)$) of graph $M_2$, where $M_2$ is the two rounds MST graph.

$$NN_j = \{\ \mu_l \mid \forall(\mu_j, \mu_l) \in E(M_2)\ \} \tag{3}$$

Furthermore, the set of two-hops neighbors ($NN_j^2$) of $\mu_j$ is the union of all the neighbors ($NN_j$) of $\mu_j$ and the set of neighbors of its own neighbors, i.e.,

$$NN_j^2 = NN_j \cup \left( \bigcup_{\mu_l \in NN_j} NN_l \right) \tag{4}$$

*3.2.2 Approximate nearest representatives of data points.* Every data point ($x_i$) is associated with a representative point ($\mu_j$), i.e., the center of the partition ($P_j$) to which $x_i$ belongs to. So, we can utilize the local neighbors of $\mu_j$ to find the nearest representatives of all the data points belong to the partition $P_j$.

*Definition 3.2 (Nearest representatives ($NR_i$) of a data point).* Let a data point $x_i$ belongs to a partition $P_j$ is represented by the center ($\mu_j$) of $P_j$, $NN_j^2$ denotes the two-hops neighbors of $\mu_j$, and $k$ is the number of neighbors of $\mu_j$, i.e., $k = |NN_j|$. Then the set of nearest representatives ($NR_i$) of $x_i$ is the $k$-representatives in $NN_j^2$ that are closest to $x_i$.

**Algorithm 2:** Bipartite graph construction

**Data:** Dataset $X$, representatives $R$, and partitions $P$.
**Result:** A bipartite proximity graph $B$ of $X$ and $R$.

1 Compute two rounds of MST of $R$. // $O(N)$ time
  Find the MST ($T_1$) of the complete graph ($K_p = \{R, E_p, W_p\}$) of $R$.
  Exclude the edges of $T_1$ from $K_p$. i.e., $E_p \leftarrow E_p \setminus E(T_1)$, and determine the second MST ($T_2$) of $K_p$.
  Combine the edges of $T_1$ and $T_2$ to form the neighborhood graph $M_2$ of $R$.
2 Find the neighbors of $R$. // $O(N^{1/2})$ time
  For every representative $\mu_j \in R$, determine the set of neighbors $NN_j$ according to Eq. (3).
  Compute the two-hops neighbors $NN_j^2$ of each representative $\mu_j$ using Eq. (4).
3 Locate the nearby representatives of objects. // $O(N)$ time
  **foreach** *partition* $P_j \in P$ **do**
    **foreach** *object* $x_i \in P_j$ **do**
      Find the number of neighbors of $\mu_j$ as $k \leftarrow |NN_j|$. /* $\mu_j$ is the center of $P_j$ */
      Determine the nearby representatives ($NR_i$) of $x_i$ by searching the $k$-closest representatives of $x_i$ among $NN_j^2$ using Definition 3.2.
4 Building the bipartite graph. // $O(N)$ time
  Construct the sparse bipartite graph $B = \{V, E, W\}$ using $NR_i$ of each $x_i \in X$, s.t., $V = X \cup R$ and every edge $(x_i, \mu_l)$ of $E$ connect a data object ($x_i$) to its one of the nearby representatives ($\mu_l \in NR_i$).
  Construct the the affinity sub-matrix $W$ of $B$ using Eq. (5).
5 **return** $B$

Suppose, we want to determine the nearby representatives of a data point $x_1$, as illustrated in Fig. 3b. So first, we find the neighbors of $\mu_1$ because $x_1$ belongs to the partition $P_1$. However, only determining the neighbors of $\mu_1$ is insufficient, like $\mu_2$ should also be one of the nearest representatives of $x_1$, but it is not a neighbor of $\mu_1$. Hence, SBSC also computes the two-hops neighbors of $\mu_1$ as explained in Definition 3.1 and illustrated in Fig. 3c. Fig. 3b clearly suggests that $\mu_1$ has five neighbors, so 5-closest representatives of $x_1$ are determined among the two-hops neighbors of $\mu_1$. Fig. 3d shows the nearest representatives of $x_1$ using the proposed technique.

*3.2.3 Affinity sub-matrix computation.* The edge set $E$ of the proposed bipartite graph ($B$) contains the edges that connect each data point to its nearby representatives based on the Definition 3.2. We represent the proximity between the data points and their representatives using an affinity sub-matrix $W \in \mathbb{R}^{N \times p}$. The Gaussian kernel [41] defines the affinity ($W_{il}$) between a data point ($x_i$) and a representative ($\mu_l$) as:

$$W_{il} = \begin{cases} \exp\left(-(d(x_i, \mu_l))^2/(2\sigma^2)\right), & \text{if } (x_i, \mu_l) \in E \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

where, $d(x_i, \mu_l)$ is the Euclidean distance between $x_i$ and $\mu_l$, and $\sigma$ is the average of distances between the data points and their nearby representatives, i.e.,

$$\sigma = \left( \sum_{x_i \in X} \sum_{\mu_l \in NR_i} d(x_i, \mu_l) \right) / \sum_{x_i \in X} |NR_i|$$

Here, $NR_i$ represents the set of nearby representatives of a data point $x_i$ as given in Definition 3.2.

Algorithm 2 describes the stepwise procedure to build the proposed bipartite proximity graph between $X$ and $R$ using the partitions $P$ of $X$.

### 3.3 Large-scale spectral clustering

In this phase, SBSC applies SC to the computed bipartite similarity graph $B$. Most of the entries in the affinity matrix ($W$) are zero, so we stored $W$ in a sparse matrix form to reduce the memory space. SBSC employs the Lancsoz method [10] in the Transfer-cuts [26] algorithm to efficiently transform the original data into the Eigen-space for the clustering task.

*3.3.1 Eigenvectors projection using the Transfers-cuts.* The standard SC solves the generalized Eigen-problem of Eq. (2) to compute the $K$ generalized eigenvectors ($f$) corresponding to the bottom $K$-eigenvalues ($\gamma$). However, solving Eq. (2) for the bipartite proximity graph carrying $(N + p)$ nodes is computationally expensive because it takes $O((N + p)^3)$ time. To improve the computational time, Li et al. [26] introduced the Transfer-cuts (Tcuts) method that utilizes the affinity sub-matrix ($W$) for the Eigen-decomposition step. Since $W$ is a non-square matrix, Tcuts reduces $W$ to the square matrix $W_R$ as follows:

$$W_R \coloneqq W^T D_X^{-1} W \tag{6}$$

Here, $W_R$ is assumed as the affinity matrix of the smaller graph whose vertices are representative points only, and $D_X$ is the diagonal matrix, s.t., the value at a position $(i, i)$ is the sum of $i$-th row of $W$, where $1 \leq i \leq N$.

The next step finds the $K$-eigenvectors ($v$) corresponding to the $K$-smallest eigenvalues ($\lambda$) of the reduced generalized Eigen-problem,

$$L_R v = \lambda D_R v \tag{7}$$

where $D_R$ is the degree matrix of $W_R$ and $L_R \coloneqq D_R - W_R$. Finding $v$ using the conventional method will take $O(p^3)$ time. The proposed method further improves the running time by considering the sparseness of $W_R$.

The generalized Eigen-problem of Eq. (7) can be transformed into a standard Eigen-problem as follows [39],

$$D_R^{-1/2} L_R D_R^{-1/2} z = \lambda z \tag{8}$$

s.t., $z = D_R^{1/2} v$. In our case, the sub-affinity matrix $W$ is highly sparse, depicting that the matrices $W_R$, $L_R$, and $D_R^{-1/2} L_R D_R^{-1/2}$ are also likely to posses sparseness. So, we solve Eq. (8) using an efficient algorithm known as the Lancsoz method to identify the $K$ standard eigenvectors ($z$) corresponding to the $K$-smallest eigenvalues of $D_R^{-1/2} L_R D_R^{-1/2}$, which subsequently helps to find the $K$ generalized eigenvectors $v = [v_1, v_2, \cdots v_K]$, s.t., $v_i$ is the $i$-th generalized eigenvector corresponding to the $i$-th smallest eigenvalue ($\lambda_i$) of the reduced Eigen-problem defined in Eq. (7). Li et al. [26] suggested that we can compute the $i$-th smallest generalized eigenvalue ($\gamma_i$) and its eigenvector ($f_i$) of Eq. (2) using $\lambda_i$ and $v_i$, respectively, which are as follows:

$$\gamma_i(2 - \gamma_i) = \lambda_i, \; f_i = \begin{bmatrix} u_i \\ v_i \end{bmatrix}, \; u_i = \frac{1}{1 - \gamma_i} D_X^{-1} W v_i \tag{9}$$

**Algorithm 3:** Eigen-decomposition using Tcuts [26] and Lancsoz methods [10]

**Data:** Dataset $X$ and affinity sub-matrix $W$.
**Result:** Eigen-space matrix $U$.

1 Reduce the sub-matrix $W$. // $O(N)$ time
  Compute the diagonal matrix $D_X$, s.t., its each diagonal element at $(i, i)$ index is computed as $\sum_{j=1 \text{ to } p} W_{ij}$
  Compute the reduced affinity matrix as $W_R \leftarrow W^T D_X^{-1} W$
2 Compute the smallest eigenvectors. // $O(KN + K^2N^{1/2})$ time
  Determine the graph Laplacian ($L_R$) of $W_R$ as $L_R \leftarrow D_R - W_R$, where $D_R$ is the degree matrix of $W_R$.
  Find $K$-eigenvectors ($z = [z_1, z_2, \cdots, z_K]$) corresponding to the $K$-smallest eigenvalues ($\lambda$) of $D_R^{-1/2} L_R D_R^{-1/2}$ using the Lancsoz method.
  Compute the $K$-smallest generalized eigenvectors ($v = [v_1, v_2, \cdots, v_K]$) of Eq. (7) using $z$, s.t., $v \leftarrow D_R^{-1/2} z$.
  Find the $K$-generalized eigenvectors ($f = [f_1, f_2, \cdots, f_K]$) of Eq. (2) by considering $v$, $D_x$, and $W$ using Eq. (9).
3 Build the Eigen-space. // $O(KN)$ time
  Stack each generalized eigenvector $f_i$ into a matrix as a column to construct the Eigen-space $U \in \mathbb{R}^{N \times K}$.
4 **return** $U$

Here, $D_X^{-1}W$ represents the transition probability matrix. Now, the first $N$ values of each vector $f_i$ are stacked in to the $i$-th column of a matrix $U \in \mathbb{R}^{N \times K}$, where $U$ represents the new Eigen-space of the input data $X$. Algorithm 3 describes each step of computing the Eigen-space $U$ using the Tcuts and Lancsoz methods.

*3.3.2 Post clustering.* Fig. 1e visualizes the heat-map of $U$ for the input instances, indicating that it is more feasible to detect the clusters of $X$ by considering the new space $U$. So, the final step applies the $K$-means algorithm on $U$ to get the clusters. Fig. 1f shows the clusters discovered by SBSC for the given dataset. Algorithm 4 describes the steps of the proposed clustering technique.

**Algorithm 4:** Proposed spectral clustering method (SBSC)

**Data:** Dataset $X$ with $K$ number of clusters.
**Result:** Clusters of $X$.

1 Divide $X$ into the set of partitions $P$ using the Algorithm 1, to get the set of representatives $R$, s.t., every representative $\mu_j \in R$ is the centroid of the partition $P_j \in P$. // $O(N \lg N)$ time
2 Construct the bipartite graph $B = \{V, E, W\}$ using Algorithm 2. // $O(N)$ time
3 Apply the Transfer-cuts along with the Lancsoz method on $W$ to construct the Eigen-space $U$ using Algorithm 3. // $O(KN + K^2N^{1/2})$ time
4 Run the $K$-means algorithm on $U$ by assuming each row of $U$ as a data instance to get the final clusters $C = \{C_1, C_2, \cdots C_K\}$. // $O(K^2Nt)$ time, where $t$ is the number of iterations of $K$-means
5 **return** $C$

## 4 Performance analysis

This section discusses the computational costs of SBSC in terms of the overall running time and graph size. The computational time of a SC method primarily depends upon the time taken to construct the graph and to find the $K$-eigenvectors. Since the proposed proximity graph represents the affinity matrix in a sparse form, therefore, the computational time to find the eigenvectors relies upon the number of edges present in the graph. Following is the analysis of the number of nodes and edges of the proposed bipartite proximity graph ($B$).

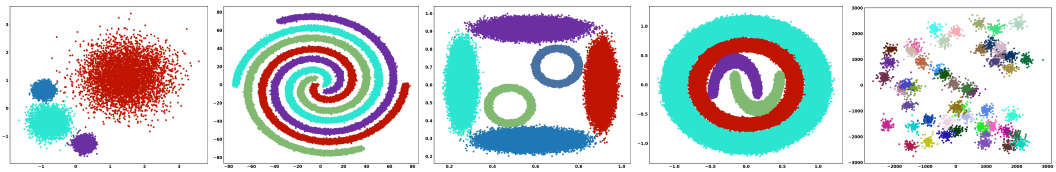
### 4.1 Size of the proximity graph

The total number of partitions in stage-1 is $O(N^{1/2}/k')$. Hence, further dividing each partition of stage-1 into $O(k')$ number of sub-partitions using the $K$-means produces $p = O(N^{1/2}/k' \times k') = O(N^{1/2})$ number of partitions and subsequently equal number of representatives. So, the total number of nodes in $B$ is $O(N + N^{1/2}) = O(N)$, and the number of edges in $B$ depends on the edges of $M_2$, i.e., the two rounds MST graph of $R$. In $M_2$, each representative point has two neighbors on an average, which means every data point is connected with an average of two representatives, resulting in the total number of edges in $B$ as $O(2 \cdot N)$. Therefore, the overall size of $B$ is $O(N)$, which is better than the existing methods mentioned in Table 1. The overall space complexity of SBSC is also linear, i.e., $O(N)$.

### 4.2 Computational time

Algorithms 1-4 mention the time complexities of the steps of SBSC. The stage-1 partitioning using Bi-means takes $O(N \lg N)$ computational time to generate $O(N^{1/2}/k')$ number of partitions. The stage-2 partitioning applies $K$-means to each of the partition in stage-1 by utilizing the Bi-means for initializing the centers, resulting in a total $O(\frac{N^{1/2}}{k'}(k'N^{1/2}\lg(k'N^{1/2}) + k'N^{1/2} \times k')$ computations. Since, $k' \ll N$, so finding the representatives using the proposed partitioning strategy takes $O(N \lg N)$ time. The total number of representatives is $O(N^{1/2})$, so the time to compute neighbors and two-hops neighbors of the representative points using $M_2$ of the complete graph of $R$ is $O(N)$. If a representative point ($\mu_j$) has $k$ neighbors, then the maximum possible number of two-hops neighbors of $\mu_j$ is $(k + k^2)$. But $k \ll N$ because every representative has on an average only two neighbors. Therefore, the running time to find the nearest representatives of the data instances by searching over the two-hops neighbors is also $O(N)$.

$W$ is a sparse matrix of linear size, so computing $W_R$ takes $O(N)$ execution time. Applying the Lancsoz method to get $K$-smallest eigenvectors of a $(p \times p)$ sparse matrix requires $O(Kp^2 + K^2p)$ running time in the worst case [28]. In our case, $p = O(N^{1/2})$, so, the total running time to construct $U$ from $W$ using the Transfer-cuts is $O(KN + K^2N^{1/2})$, and applying $K$-means for $t$ number of



(a) (b) (c) (d) (e)

Fig. 4. Large sized synthetic datasets: (a) Varied-variances. (b) Spirals. (c) Blobs-Rings. (d) Circles-Moons. (e) Noisy-Blobs-50.

Table 2. Details of synthetic and real datasets used for the experiment.

| Dataset | N | F | K |
|---|---|---|---|
| DS1: Varied-variances | 20000 | 2 | 4 |
| DS2: Spirals | 320000 | 2 | 4 |
| DS3: Blobs-Rings | 1000000 | 2 | 6 |
| DS4: Circles-Moons | 10000000 | 2 | 4 |
| DS5: Noisy-Blobs-50 | 10000 | 2 | 50 |
| DR1: Occupancy | 10808 | 5 | 2 |
| DR2: Pendigits | 10992 | 16 | 10 |
| DR3: Letter | 20000 | 16 | 26 |
| DR4: Shuttle | 43500 | 9 | 7 |
| DR5: Gas-sensor | 13910 | 129 | 6 |
| DR6: ISOLET | 7797 | 617 | 26 |
| DR7: Smartphones | 10929 | 561 | 12 |
| DR8: Covtype | 581012 | 54 | 7 |
| DR9: MNIST | 70000 | 784 | 10 |
| DR10: HELENA | 65196 | 27 | 100 |
| DR11: Winnipeg | 325834 | 174 | 7 |
| DR12: Dionis | 416188 | 60 | 355 |
| DR13: EMNIST | 240000 | 784 | 10 |

iterations on $U$ takes $O(K^2Nt)$ computational time. Here, $t \ll N$, therefore, overall running time of SBSC is $O(N(K^2 + \lg N))$, ignoring the number of features.

## 5 Experimental results and analyses

This section empirically compares the clustering performance of SBSC with a few benchmark and state-of-the-art SC algorithms. The competing methods are $K$-means [16, 27], Agglomerative Clustering using complete linkage (CL [18]), DBSCAN [8], fast search and density peaks (FDP [37]), a traditional SC algorithm with $k$NN as a similarity graph ($k$NN-SC [41]), two ASC methods (KASP [45] and DCDP-ASC [4]), a recent granular ball-based spectral clustering (GBSC) algorithm [43], and four BSC methods (LSC-R, LSC-K [2], USPEC [13], and DnC [23]).

$k$NN-SC requires the number of neighbors $k$ as a sparsity parameter, which is taken as $k = \log(N)$, as suggested in [41], whereas DBSCAN tunes two external parameters ($\epsilon$ and $min - pnts$), which we vary as $\epsilon = [0.1, 0.3, 0.5, 0.7, 0.9, 1, 3, 5, 7, 9]$ and $min - pnts = [3, 5, 7, 9]$. The cut-off factor of FDP is taken as 2%. The methods LSC-R, LSC-K, USPEC, and DnC have two external parameters, i.e., the number of representative points ($p$) and nearest representatives ($k$) respectively. For a fair comparison, we set the values of $p = 1000$ and $k = 5$ as suggested by DnC. KASP is dependent upon two parameters - $p$ and $\sigma$. We keep the parameter $p$ same as other methods, i.e., 1000. $\sigma$ is the sparsity parameter that is used in the Gaussian kernel, which we set as the average of all edge weights of the graph, as suggested in [41]. DCDP-ASC is a recent ASC technique that takes only $\alpha$ as an input parameter, which decides $p$. Generally, $\alpha$ is assumed to be zero, but in the case of larger datasets, it is taken as a non-zero floating point value [4]. GBSC requires tuning a sparsity parameter $\sigma$, which is set from 0.2 to 1.0.

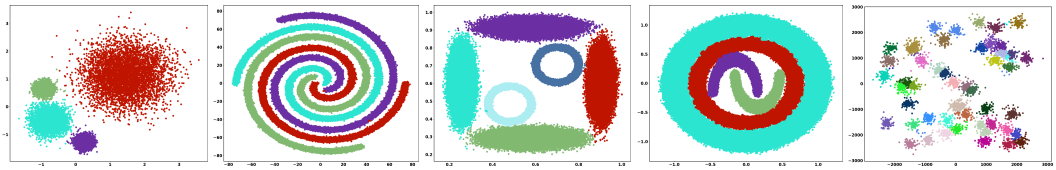
All methods are implemented using Python 3.10 and executed on Intel(R) Xeon(R) Gold 2.10 GHz processor with 128GB RAM and Windows 10 (64 bit) OS. We use Scikit-learn [31] library's in-built function to execute $K$-means, CL, and DBSCAN algorithms. The Python code of GBSC is taken from the link shared in its paper [43]. We have incorporated standard libraries like NumPy, SciPy, Pandas, Matplotlib, and Scikit-learn into the Python codes of SBSC and other techniques to make the implementation efficient. Also, we have implemented the code for finding MST in the

Cython environment, an optimizing compiler that enables writing C extensions for Python. The code of SBSC is available at [20]. The correctness of clusters is measured based on two conventional validity indices (VI), viz. clustering accuracy (CA) and rand index (RI) [34, 44]. Both validity indices vary from 0 to 1, where 1 means perfect clustering. The results of the methods vary on each run for a given dataset, therefore, the average results of ten runs are considered.

### 5.1 Datasets

The experiment includes five two-dimensional synthetic and thirteen diversified real datasets respectively. Fig. 4 visualizes the clusters of synthetic datasets with the ground truth labels. The Varied variances dataset consists of four touching blobs with noisy instances and different densities. Performing the clustering on Spirals and Circles-Moons datasets is challenging because of the presence of non-convex-shaped clusters and clusters inside a cluster. The Noisy-Blobs-50 dataset have a higher number of clusters, i.e., $K = 50$, and the Circles-Moons dataset carries the highest number of instances, i.e., ten million.

Table 2 shows details of all the large-sized synthetic (DS1-DS5) and real-world (DR1-DR13) datasets, available at [20] and [6, 7], respectively. The real datasets are distinctive in terms of the number of data instances ($N$), dimensions ($F$), and clusters ($K$), respectively. The datasets DR5-DR7, DR9, DR11, and DR13 possess high number of attributes, and datasets DR8, & DR11-DR13 have a large number of data instances. Also, for datasets DR3, DR6, DR10, and DR12, the number of actual clusters is comparatively more. We have also included the datasets (DR4, DR7, DR8, DR10, and DR11) with unbalanced clusters in terms of clusters' size.



(a) (b) (c) (d) (e)

Fig. 5. Visualization of clusters produced by SBSC: (a) Varied-variances. (b) Spirals. (c) Blobs-Rings. (d) Circles-Moons. (e) Noisy-Blobs-50

### 5.2 Clustering quality analyses

Table 3 shows the clustering quality of the competing methods for the above-discussed datasets based on CA and RI. The symbol (±) separates each entry of Table 3 into two values, s.t., the values at the left and right of '±' indicate the mean and standard deviation of CA / RI values of ten runs of the given method for a dataset, respectively. The last two rows indicate the average scores of CA(s) and RI(s) produced by the clustering methods for all the datasets respectively. In the table, a few entries are filled with 'N/A', which indicates that the method could not produce the clusters for the given dataset because of the out-of-memory error. Also, Fig. 5 visualizes the best results of SBSC for each synthetic dataset in terms of CA. The results indicate the following observations regarding the performance of SBSC.

- Fig. 5 suggests that SBSC is capable of detecting complex-shaped clusters like touching clusters with noises & varying densities, non-convex shaped clusters, and clusters inside a cluster. It is unable to detect clusters properly for DS5 because of the large number of clusters present in it but it performs best among all the competing methods based on Table 3.

Table 3. CA and RI of clusters discovered by the competing clustering techniques for the given synthetic and real datasets.

| Dataset | VI | $K$-means | CL | DBSCAN | FDP | $k$NN-SC | KASP | DCDP-ASC | GBSC | LSC-R | LSC-K | USPEC | DnC | SBSC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DS1 | CA | $0.882_{\pm 0.16}$ | $0.389_{\pm 0.00}$ | $0.748_{\pm 0.00}$ | $0.970_{\pm 0.00}$ | $0.993_{\pm 0.00}$ | $\mathbf{0.999}_{\pm 0.00}$ | $0.629_{\pm 0.00}$ | $\mathbf{0.999}_{\pm 0.00}$ | $0.998_{\pm 0.00}$ | $\mathbf{0.999}_{\pm 0.00}$ | $0.974_{\pm 0.07}$ | $\mathbf{0.999}_{\pm 0.00}$ | $\mathbf{0.999}_{\pm 0.00}$ |
| | RI | $0.941_{\pm 0.07}$ | $0.540_{\pm 0.00}$ | $0.873_{\pm 0.00}$ | $0.971_{\pm 0.00}$ | $0.990_{\pm 0.00}$ | $\mathbf{0.999}_{\pm 0.00}$ | $0.843_{\pm 0.00}$ | $\mathbf{0.999}_{\pm 0.00}$ | $0.998_{\pm 0.00}$ | $\mathbf{0.999}_{\pm 0.00}$ | $0.986_{\pm 0.04}$ | $\mathbf{0.999}_{\pm 0.00}$ | $\mathbf{0.999}_{\pm 0.00}$ |
| DS2 | CA | $0.330_{\pm 0.00}$ | N/A | N/A | N/A | N/A | $0.354_{\pm 0.02}$ | $0.266_{\pm 0.00}$ | $0.250_{\pm 0.00}$ | $0.717_{\pm 0.07}$ | $0.999_{\pm 0.00}$ | $0.995_{\pm 0.01}$ | $0.990_{\pm 0.02}$ | $\mathbf{1.000}_{\pm 0.00}$ |
| | RI | $0.635_{\pm 0.00}$ | N/A | N/A | N/A | N/A | $0.609_{\pm 0.01}$ | $0.578_{\pm 0.00}$ | $0.250_{\pm 0.00}$ | $0.768_{\pm 0.05}$ | $0.999_{\pm 0.00}$ | $0.995_{\pm 0.01}$ | $0.991_{\pm 0.02}$ | $\mathbf{1.000}_{\pm 0.00}$ |
| DS3 | CA | $0.732_{\pm 0.09}$ | N/A | N/A | N/A | N/A | $0.834_{\pm 0.10}$ | $0.250_{\pm 0.00}$ | N/A | $0.917_{\pm 0.06}$ | $0.917_{\pm 0.04}$ | $0.999_{\pm 0.00}$ | $0.992_{\pm 0.02}$ | $\mathbf{1.000}_{\pm 0.00}$ |
| | RI | $0.908_{\pm 0.03}$ | N/A | N/A | N/A | N/A | $0.949_{\pm 0.02}$ | $0.742_{\pm 0.00}$ | N/A | $0.978_{\pm 0.02}$ | $0.978_{\pm 0.01}$ | $\mathbf{1.000}_{\pm 0.00}$ | $0.997_{\pm 0.01}$ | $\mathbf{1.000}_{\pm 0.00}$ |
| DS4 | CA | $0.484_{\pm 0.00}$ | N/A | N/A | N/A | N/A | $0.880_{\pm 0.01}$ | $0.112_{\pm 0.00}$ | N/A | $\mathbf{1.000}_{\pm 0.00}$ | $\mathbf{1.000}_{\pm 0.00}$ | $\mathbf{1.000}_{\pm 0.00}$ | $\mathbf{1.000}_{\pm 0.00}$ | $\mathbf{1.000}_{\pm 0.00}$ |
| | RI | $0.675_{\pm 0.00}$ | N/A | N/A | N/A | N/A | $0.909_{\pm 0.01}$ | $0.114_{\pm 0.00}$ | N/A | $\mathbf{1.000}_{\pm 0.00}$ | $\mathbf{1.000}_{\pm 0.00}$ | $\mathbf{1.000}_{\pm 0.00}$ | $\mathbf{1.000}_{\pm 0.00}$ | $\mathbf{1.000}_{\pm 0.00}$ |
| DS5 | CA | $0.752_{\pm 0.04}$ | $0.843_{\pm 0.00}$ | $0.614_{\pm 0.00}$ | $0.441_{\pm 0.00}$ | $0.839_{\pm 0.00}$ | $0.217_{\pm 0.04}$ | $0.207_{\pm 0.00}$ | $0.841_{\pm 0.00}$ | $0.772_{\pm 0.01}$ | $0.777_{\pm 0.01}$ | $0.810_{\pm 0.01}$ | $0.838_{\pm 0.02}$ | $\mathbf{0.850}_{\pm 0.01}$ |
| | RI | $0.988_{\pm 0.00}$ | $0.992_{\pm 0.00}$ | $0.085_{\pm 0.00}$ | $0.935_{\pm 0.00}$ | $0.992_{\pm 0.00}$ | $0.867_{\pm 0.01}$ | $0.989_{\pm 0.00}$ | $0.992_{\pm 0.00}$ | $0.960_{\pm 0.01}$ | $0.974_{\pm 0.00}$ | $0.992_{\pm 0.00}$ | $0.991_{\pm 0.00}$ | $\mathbf{0.993}_{\pm 0.00}$ |
| DR1 | CA | $0.943_{\pm 0.00}$ | $0.750_{\pm 0.00}$ | $0.830_{\pm 0.00}$ | $0.958_{\pm 0.00}$ | $0.503_{\pm 0.00}$ | $0.647_{\pm 0.09}$ | $0.728_{\pm 0.00}$ | $0.815_{\pm 0.00}$ | $0.965_{\pm 0.01}$ | $0.936_{\pm 0.09}$ | $0.936_{\pm 0.09}$ | $0.864_{\pm 0.11}$ | $\mathbf{0.984}_{\pm 0.00}$ |
| | RI | $0.892_{\pm 0.00}$ | $0.625_{\pm 0.00}$ | $0.870_{\pm 0.00}$ | $0.945_{\pm 0.00}$ | $0.500_{\pm 0.00}$ | $0.560_{\pm 0.05}$ | $0.604_{\pm 0.00}$ | $0.699_{\pm 0.00}$ | $0.932_{\pm 0.02}$ | $0.898_{\pm 0.14}$ | $0.898_{\pm 0.14}$ | $0.791_{\pm 0.17}$ | $\mathbf{0.969}_{\pm 0.00}$ |
| DR2 | CA | $0.700_{\pm 0.05}$ | $0.553_{\pm 0.00}$ | $0.493_{\pm 0.00}$ | $0.380_{\pm 0.00}$ | $0.848_{\pm 0.03}$ | $0.432_{\pm 0.04}$ | $0.801_{\pm 0.00}$ | $0.609_{\pm 0.03}$ | $0.799_{\pm 0.03}$ | $0.810_{\pm 0.03}$ | $0.842_{\pm 0.03}$ | $0.831_{\pm 0.03}$ | $\mathbf{0.858}_{\pm 0.02}$ |
| | RI | $0.915_{\pm 0.01}$ | $0.877_{\pm 0.00}$ | $0.864_{\pm 0.00}$ | $0.764_{\pm 0.00}$ | $0.952_{\pm 0.01}$ | $0.799_{\pm 0.03}$ | $0.940_{\pm 0.00}$ | $0.89_{\pm 0.01}$ | $0.935_{\pm 0.01}$ | $0.942_{\pm 0.01}$ | $0.949_{\pm 0.01}$ | $0.947_{\pm 0.01}$ | $\mathbf{0.954}_{\pm 0.01}$ |
| DR3 | CA | $0.255_{\pm 0.01}$ | $0.203_{\pm 0.00}$ | $0.247_{\pm 0.00}$ | $0.072_{\pm 0.00}$ | $0.243_{\pm 0.02}$ | $0.109_{\pm 0.02}$ | $0.325_{\pm 0.00}$ | $0.132_{\pm 0.00}$ | $0.297_{\pm 0.01}$ | $0.319_{\pm 0.01}$ | $0.341_{\pm 0.01}$ | $0.343_{\pm 0.01}$ | $\mathbf{0.348}_{\pm 0.00}$ |
| | RI | $0.931_{\pm 0.00}$ | $0.905_{\pm 0.00}$ | $0.717_{\pm 0.00}$ | $0.141_{\pm 0.00}$ | $0.830_{\pm 0.03}$ | $0.727_{\pm 0.04}$ | $0.907_{\pm 0.00}$ | $0.923_{\pm 0.00}$ | $0.914_{\pm 0.00}$ | $0.919_{\pm 0.01}$ | $0.928_{\pm 0.00}$ | $\mathbf{0.933}_{\pm 0.00}$ | $\mathbf{0.933}_{\pm 0.00}$ |
| DR4 | CA | $0.624_{\pm 0.10}$ | $0.784_{\pm 0.00}$ | $0.906_{\pm 0.00}$ | $0.784_{\pm 0.00}$ | $0.484_{\pm 0.02}$ | $0.462_{\pm 0.10}$ | $0.779_{\pm 0.00}$ | $0.825_{\pm 0.00}$ | $0.697_{\pm 0.04}$ | $0.843_{\pm 0.08}$ | $0.836_{\pm 0.06}$ | $0.836_{\pm 0.04}$ | $\mathbf{0.908}_{\pm 0.01}$ |
| | RI | $0.608_{\pm 0.06}$ | $0.642_{\pm 0.00}$ | $0.866_{\pm 0.00}$ | $0.642_{\pm 0.00}$ | $0.537_{\pm 0.02}$ | $0.508_{\pm 0.05}$ | $0.784_{\pm 0.00}$ | $0.749_{\pm 0.00}$ | $0.691_{\pm 0.05}$ | $0.781_{\pm 0.09}$ | $0.739_{\pm 0.09}$ | $0.749_{\pm 0.06}$ | $\mathbf{0.893}_{\pm 0.04}$ |
| DR5 | CA | $0.395_{\pm 0.00}$ | $0.311_{\pm 0.00}$ | $0.216_{\pm 0.00}$ | $0.242_{\pm 0.00}$ | $0.216_{\pm 0.00}$ | $\mathbf{0.490}_{\pm 0.02}$ | $0.285_{\pm 0.00}$ | $0.233_{\pm 0.01}$ | $0.426_{\pm 0.03}$ | $0.388_{\pm 0.07}$ | $0.336_{\pm 0.05}$ | $0.352_{\pm 0.08}$ | $0.427_{\pm 0.00}$ |
| | RI | $0.716_{\pm 0.01}$ | $0.421_{\pm 0.00}$ | $0.176_{\pm 0.00}$ | $0.219_{\pm 0.00}$ | $0.176_{\pm 0.00}$ | $\mathbf{0.755}_{\pm 0.01}$ | $0.352_{\pm 0.00}$ | $0.512_{\pm 0.07}$ | $0.690_{\pm 0.07}$ | $0.594_{\pm 0.17}$ | $0.459_{\pm 0.14}$ | $0.518_{\pm 0.17}$ | $0.717_{\pm 0.00}$ |
| DR6 | CA | $0.533_{\pm 0.02}$ | $0.417_{\pm 0.00}$ | $0.332_{\pm 0.00}$ | $0.067_{\pm 0.00}$ | $\mathbf{0.586}_{\pm 0.02}$ | $0.180_{\pm 0.02}$ | $0.472_{\pm 0.00}$ | $0.256_{\pm 0.00}$ | $0.528_{\pm 0.02}$ | $0.525_{\pm 0.02}$ | $0.573_{\pm 0.02}$ | $0.575_{\pm 0.01}$ | $0.575_{\pm 0.01}$ |
| | RI | $0.958_{\pm 0.00}$ | $0.928_{\pm 0.00}$ | $0.783_{\pm 0.00}$ | $0.142_{\pm 0.00}$ | $\mathbf{0.963}_{\pm 0.00}$ | $0.699_{\pm 0.01}$ | $0.953_{\pm 0.00}$ | $0.926_{\pm 0.00}$ | $0.952_{\pm 0.00}$ | $0.952_{\pm 0.00}$ | $0.960_{\pm 0.00}$ | $0.962_{\pm 0.00}$ | $0.960_{\pm 0.00}$ |
| DR7 | CA | $0.515_{\pm 0.03}$ | $0.412_{\pm 0.00}$ | $0.291_{\pm 0.00}$ | $0.340_{\pm 0.00}$ | $0.632_{\pm 0.01}$ | $0.460_{\pm 0.03}$ | $0.576_{\pm 0.00}$ | $0.358_{\pm 0.00}$ | $0.620_{\pm 0.03}$ | $0.651_{\pm 0.04}$ | $0.629_{\pm 0.02}$ | $0.640_{\pm 0.04}$ | $\mathbf{0.655}_{\pm 0.02}$ |
| | RI | $0.866_{\pm 0.01}$ | $0.750_{\pm 0.00}$ | $0.786_{\pm 0.00}$ | $0.653_{\pm 0.00}$ | $0.886_{\pm 0.00}$ | $0.764_{\pm 0.01}$ | $0.881_{\pm 0.00}$ | $0.788_{\pm 0.01}$ | $0.885_{\pm 0.01}$ | $0.892_{\pm 0.01}$ | $0.889_{\pm 0.01}$ | $\mathbf{0.892}_{\pm 0.01}$ | $\mathbf{0.894}_{\pm 0.00}$ |
| DR8 | CA | $0.251_{\pm 0.00}$ | N/A | N/A | N/A | N/A | $0.355_{\pm 0.03}$ | $\mathbf{0.397}_{\pm 0.00}$ | N/A | $0.298_{\pm 0.01}$ | $0.305_{\pm 0.01}$ | $0.266_{\pm 0.01}$ | $0.276_{\pm 0.01}$ | $0.267_{\pm 0.01}$ |
| | RI | $0.576_{\pm 0.00}$ | N/A | N/A | N/A | N/A | $0.581_{\pm 0.01}$ | $0.465_{\pm 0.00}$ | N/A | $0.578_{\pm 0.00}$ | $0.579_{\pm 0.00}$ | $0.582_{\pm 0.00}$ | $0.579_{\pm 0.00}$ | $\mathbf{0.589}_{\pm 0.00}$ |
| DR9 | CA | $0.547_{\pm 0.04}$ | $0.322_{\pm 0.00}$ | $0.113_{\pm 0.00}$ | $0.113_{\pm 0.00}$ | $0.739_{\pm 0.00}$ | $0.353_{\pm 0.02}$ | $0.636_{\pm 0.00}$ | $0.379_{\pm 0.00}$ | $0.642_{\pm 0.02}$ | $0.750_{\pm 0.02}$ | $0.765_{\pm 0.02}$ | $\mathbf{0.783}_{\pm 0.01}$ | $0.772_{\pm 0.00}$ |
| | RI | $0.884_{\pm 0.01}$ | $0.761_{\pm 0.00}$ | $0.611_{\pm 0.00}$ | $0.101_{\pm 0.00}$ | $0.934_{\pm 0.00}$ | $0.681_{\pm 0.01}$ | $0.910_{\pm 0.00}$ | $0.852_{\pm 0.00}$ | $0.907_{\pm 0.00}$ | $0.930_{\pm 0.01}$ | $0.932_{\pm 0.00}$ | $\mathbf{0.938}_{\pm 0.00}$ | $0.935_{\pm 0.00}$ |
| DR10 | CA | $0.084_{\pm 0.00}$ | $0.107_{\pm 0.00}$ | $0.108_{\pm 0.00}$ | $0.062_{\pm 0.00}$ | $0.061_{\pm 0.00}$ | $\mathbf{0.110}_{\pm 0.00}$ | $0.103_{\pm 0.00}$ | $0.067_{\pm 0.00}$ | $0.086_{\pm 0.00}$ | $0.087_{\pm 0.00}$ | $0.086_{\pm 0.00}$ | $0.086_{\pm 0.00}$ | $0.088_{\pm 0.01}$ |
| | RI | $\mathbf{0.968}_{\pm 0.01}$ | $0.953_{\pm 0.00}$ | $0.916_{\pm 0.00}$ | $0.026_{\pm 0.00}$ | $0.023_{\pm 0.00}$ | $0.750_{\pm 0.01}$ | $0.941_{\pm 0.00}$ | $0.963_{\pm 0.00}$ | $0.967_{\pm 0.00}$ | $0.967_{\pm 0.00}$ | $\mathbf{0.968}_{\pm 0.00}$ | $\mathbf{0.968}_{\pm 0.01}$ | $\mathbf{0.968}_{\pm 0.00}$ |
| DR11 | CA | $0.358_{\pm 0.01}$ | N/A | N/A | N/A | N/A | $0.386_{\pm 0.00}$ | $0.252_{\pm 0.00}$ | $0.261_{\pm 0.01}$ | $0.353_{\pm 0.01}$ | $0.364_{\pm 0.00}$ | $0.359_{\pm 0.02}$ | $0.357_{\pm 0.01}$ | $\mathbf{0.393}_{\pm 0.01}$ |
| | RI | $\mathbf{0.729}_{\pm 0.01}$ | N/A | N/A | N/A | N/A | $0.712_{\pm 0.01}$ | $0.250_{\pm 0.00}$ | $0.209_{\pm 0.01}$ | $0.723_{\pm 0.01}$ | $0.696_{\pm 0.00}$ | $0.725_{\pm 0.00}$ | $0.722_{\pm 0.01}$ | $\mathbf{0.729}_{\pm 0.01}$ |
| DR12 | CA | $0.038_{\pm 0.00}$ | N/A | N/A | N/A | N/A | $0.030_{\pm 0.00}$ | $0.119_{\pm 0.00}$ | N/A | $0.141_{\pm 0.00}$ | $0.117_{\pm 0.00}$ | $0.146_{\pm 0.00}$ | $0.079_{\pm 0.01}$ | $\mathbf{0.147}_{\pm 0.00}$ |
| | RI | $0.886_{\pm 0.02}$ | N/A | N/A | N/A | N/A | $0.854_{\pm 0.03}$ | $0.982_{\pm 0.00}$ | N/A | $0.989_{\pm 0.00}$ | $0.993_{\pm 0.00}$ | $\mathbf{0.994}_{\pm 0.00}$ | $0.993_{\pm 0.00}$ | $0.993_{\pm 0.00}$ |
| DR13 | CA | $0.547_{\pm 0.04}$ | N/A | N/A | N/A | N/A | $0.320_{\pm 0.01}$ | $0.282_{\pm 0.00}$ | $0.100_{\pm 0.00}$ | $0.647_{\pm 0.04}$ | $0.769_{\pm 0.02}$ | $0.761_{\pm 0.02}$ | $0.786_{\pm 0.02}$ | $\mathbf{0.835}_{\pm 0.01}$ |
| | RI | $0.885_{\pm 0.01}$ | N/A | N/A | N/A | N/A | $0.697_{\pm 0.02}$ | $0.732_{\pm 0.00}$ | $0.100_{\pm 0.00}$ | $0.902_{\pm 0.01}$ | $0.931_{\pm 0.01}$ | $0.930_{\pm 0.00}$ | $0.936_{\pm 0.01}$ | $\mathbf{0.946}_{\pm 0.00}$ |
| Avg. Score | CA | 0.499 | N/A | N/A | N/A | N/A | 0.423 | 0.401 | N/A | 0.606 | 0.642 | 0.647 | 0.645 | **0.672** |
| | RI | 0.830 | N/A | N/A | N/A | N/A | 0.746 | 0.720 | N/A | 0.876 | 0.890 | 0.884 | 0.884 | **0.915** |

- For ten real datasets, i.e., DR1-DR5, DR7, DR9, and DR11-DR13, SBSC is amongst the best of two clustering methods based on both CA and RI validity indices. For DR6, SBSC performs the second best in terms of CA, and in terms of RI, SBSC performs best for DR8 & DR10.
- The experimental results on datasets like DS2-DS4, DR8, and DR11-DR13 suggest that SBSC discovers better clusters for large-sized data than the competing SC techniques. SBSC also produces higher quality of clusters than the other techniques for high-dimensional data (DR5-DR7, DR9, DR11, and DR13) and the datasets with unbalanced clusters.
- Overall, SBSC is the most effective among the competing methods based on the average scores of CA(s) and RI(s) produced for all datasets.

*5.2.1 Sensitivity with $K$.* Although SBSC produces better results for datasets (DS5, DR3, DR6, DR10, and DR12) with a high number of clusters $K$, this section analyses further to study the impact of $K$ on the clustering results of competing methods. For this, we choose, the DS5 synthetic dataset that contains fifty noisy blobs as clusters. This experiment sets checkpoints as $K = [10, 20, 30, 40, 50]$ by varying the $K$. Each clustering algorithm is applied for ten iterations for every ($K \leq 40$). In every iteration, we have randomly sampled $K$ clusters from DS5 dataset and then applied the competing clustering algorithms to the sample data.

Fig. 6 plots the clustering accuracy of SBSC and competing methods with the increase in the value of $K$. The figure demonstrates that SBSC produces better clustering quality than its competitors for each value of $K$. Though the results of SBSC are close to the CL algorithm for every checkpoint, but CL is computationally expensive because of its quadratic time complexity.

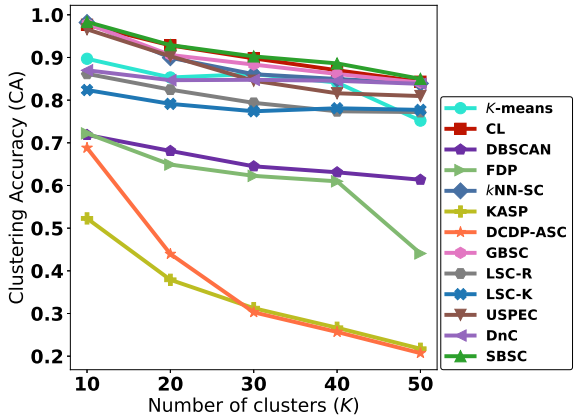



Fig. 6. Clustering accuracies produced by competing techniques by varying the number of clusters ($K$) for the DS5 dataset.

*5.2.2 Significant Testing.* We also perform the Wilcoxon rank-sum test [32] in the experiment to examine the significant difference in the clustering accuracies produced by SBSC and its competing techniques. Since the results do not vary for DS1-DS4 datasets, we excluded them from this analysis. Similarly, deterministic methods (CL, DBSCAN, FDP, and DCDP-ASC) producing the same results in each run for a given dataset are also not included. The Wilcoxon rank-sum is a non-parametric test whose null hypothesis is that the values of two sets are from a similar distribution. Table 4 shows the $p$-values between the clustering accuracies produced by SBSC and competing methods for each dataset, indicating the rejection of the null hypothesis at a 5% significance level for several cases. In other words, SBSC produces clusters significantly better than its competitors based on CA. In a few cases, ($p > 0.05$) shows no significant changes in the results of SBSC and its competitors.

Table 4. $p$-values between the clustering accuracies of SBSC and competing methods based on the Wilcoxon rank-sum test.

| Dataset | SBSC vs $K$-means | SBSC vs $k$NN-SC | SBSC vs KASP | SBSC vs GBSC | SBSC vs LSC-R | SBSC vs LSC-K | SBSC vs USPEC | SBSC vs DnC |
|---|---|---|---|---|---|---|---|---|
| DS5 | 0.000507 | 0.545350 | 0.000157 | 0.069642 | 0.000157 | 0.000157 | 0.289918 | 0.325751 |
| DR1 | 0.000157 | 0.000157 | 0.000157 | 0.000157 | 0.000157 | 0.000157 | 0.449692 | 0.023342 |
| DR2 | 0.000157 | 0.010165 | 0.000157 | 0.023342 | 0.000670 | 0.005159 | 0.049366 | 0.034294 |
| DR3 | 0.000157 | 0.000157 | 0.000157 | 0.000157 | 0.000157 | 0.000157 | 0.004072 | 0.112411 |
| DR4 | 0.000157 | 0.000157 | 0.000157 | 0.000157 | 0.000157 | 0.049366 | 0.015564 | 0.002497 |
| DR5 | 0.000157 | 0.000157 | 0.000157 | 0.000157 | 1.000000 | 0.130570 | 0.000157 | 0.023342 |
| DR6 | 0.000285 | 0.000157 | 0.000157 | 0.000157 | 0.000157 | 0.000183 | 0.705457 | 0.939743 |
| DR7 | 0.000157 | 0.130570 | 0.000157 | 0.000157 | 0.015564 | 0.449692 | 0.011330 | 0.449692 |
| DR8 | 0.002497 | N/A | 0.000157 | N/A | 0.000157 | 0.000157 | 0.545350 | 0.041250 |
| DR9 | 0.000157 | 0.130570 | 0.000157 | 0.000157 | 0.000157 | 0.002497 | 0.449692 | 0.023342 |
| DR10 | 0.006502 | 0.000157 | 0.000157 | 0.000157 | 0.049366 | 0.364346 | 0.041250 | 0.049366 |
| DR11 | 0.000881 | N/A | 0.002497 | 0.005159 | 0.000507 | 0.000881 | 0.000670 | 0.000670 |
| DR12 | 0.000157 | N/A | 0.000157 | N/A | 0.000157 | 0.000157 | 0.010165 | 0.000157 |
| DR13 | 0.000157 | N/A | 0.000157 | 0.000157 | 0.000157 | 0.000212 | 0.000157 | 0.000285 |

## 5.3 Clustering time analyses

Table 5 shows the average of execution time (in seconds) taken by the competing algorithms in ten runs to discover the final clusters of the given datasets, suggesting that SBSC is the fastest among all the competing techniques. For better comparison, we also visualize the running time

Table 5. Execution time of different SC methods in seconds for the given synthetic and real datasets.

| Dataset | $k$NN-SC | KASP | DCDP-ASC | GBSC | LSC-R | LSC-K | USPEC | DnC | SBSC |
|---|---|---|---|---|---|---|---|---|---|
| DS1 | 170.4 | 7.2 | 272.9 | 117.8 | 8.4 | 10.9 | 9.2 | 8.9 | **5.6** |
| DS2 | N/A | 66.4 | 2008.6 | 4071.2 | 75.7 | 137.3 | 56.1 | 49.0 | **22.4** |
| DS3 | N/A | 202.2 | 4259.0 | N/A | 225.4 | 416.7 | 169.2 | 133.2 | **48.6** |
| DS4 | N/A | 1940.0 | 14739.1 | N/A | 2546.6 | 3865.2 | 1638.6 | 1265.9 | **315.3** |
| DS5 | 472.1 | 5.2 | 225.3 | 38.4 | 6.3 | 9.2 | 6.2 | 7.6 | **3.2** |
| DR1 | 601.6 | 3.4 | 24.4 | 37.3 | 5.8 | 7.4 | 6.3 | 7.5 | **1.8** |
| DR2 | 121.2 | 5.3 | 102.0 | 45.9 | 8.4 | 7.8 | 7.0 | 7.6 | **1.1** |
| DR3 | 317.1 | 6.1 | 78.4 | 94.5 | 12.2 | 13.3 | 11.7 | 11.9 | **4.4** |
| DR4 | 1384.2 | 22.8 | 4783.1 | 879.5 | 15.3 | 23.3 | 12.8 | 12.7 | **10.6** |
| DR5 | 1371.6 | 10.4 | 31.1 | 71.0 | 10.1 | 12.7 | 10.1 | 7.9 | **3.2** |
| DR6 | 174.3 | 7.5 | 108.2 | 13.6 | 13.2 | 16.7 | 11.4 | 23.9 | **3.1** |
| DR7 | 198.2 | 9.6 | 498.3 | 52.7 | 16.3 | 19.4 | 15.7 | 13.7 | **3.3** |
| DR8 | N/A | 185.5 | 8262.7 | N/A | 187.9 | 358.0 | 125.8 | 76.9 | **41.2** |
| DR9 | 8258.3 | 192.6 | 5491.5 | 1253.8 | 97.0 | 281.5 | 38.3 | 47.1 | **30.6** |
| DR10 | 5767.3 | 134.0 | 483.6 | 3426.1 | 95.1 | 191.7 | 53.6 | 52.6 | **41.7** |
| DR11 | N/A | 218.6 | 8374.5 | 1415.5 | 168.0 | 373.4 | 91.8 | 98.7 | **45.1** |
| DR12 | N/A | 1147.2 | 9170.2 | N/A | 1062.1 | 1564.7 | 1394.2 | 1331.4 | **624.7** |
| DR13 | N/A | 372.9 | 1468.1 | 2526.0 | 482.7 | 593.4 | 144.8 | 153.2 | **93.5** |

by varying the number of points ($N$) of the Circles-Moons dataset for four SC techniques (SBSC, LCS-R, USPEC, DnC) that use a bipartite proximity graph. We have not included LSC-K because it takes longer execution time than LSC-R. There are ten checkpoints with varying $N$ from one million to ten million as demonstrated in Fig. 7. The heights of the bars indicate the time taken by the methods in seconds for different values of $N$, which indicates that SBSC is significantly faster than other techniques for all checkpoints. Efficiently constructing the graph using the local neighbors contributes to the fastness of SBSC mainly due to the reduced number of searches over fewer representatives to collect the neighborhood.

Another reason for the fast computation of SBSC is the high sparsity of the proposed bipartite graph, as it only contains a linear number of edges. The fewer the edges in the graph, the faster the

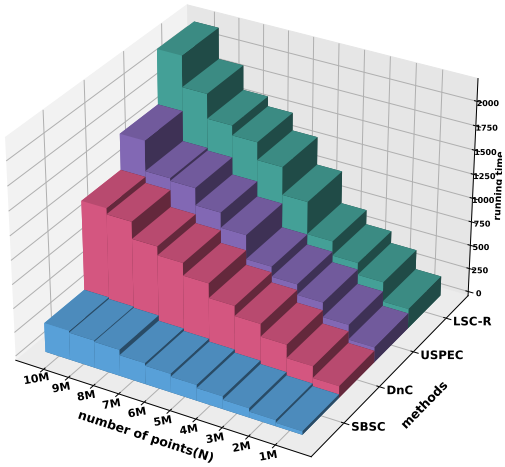



Fig. 7. Execution time (in seconds) of the clustering methods with varying $N$ for the Circles-Moons dataset.

Table 6. Number of edges present in the bipartite proximity graphs of different SC techniques for the given datasets.

| Dataset | LSC-R | LSC-K | USPEC | DnC | SBSC |
|---|---|---|---|---|---|
| DS1 | 98976 | 99990 | 99983 | 99966 | **81479** |
| DS2 | 1599000 | 1600000 | 1599983 | 1600000 | **1290672** |
| DS3 | 7498804 | 7499992 | 7499763 | 7499902 | **6319708** |
| DS4 | 49999000 | 50000000 | 49999995 | 50000000 | **40593916** |
| DS5 | 49000 | 49988 | 49990 | 49909 | **43352** |
| DR1 | 52541 | 53920 | 53830 | 53824 | **44254** |
| DR2 | 53960 | 54941 | 54945 | 54922 | **47606** |
| DR3 | 98743 | 99998 | 99977 | 99999 | **84469** |
| DR4 | 215711 | 216841 | 216742 | 216874 | **181932** |
| DR5 | 68430 | 69446 | 69474 | 69430 | **57398** |
| DR6 | 37985 | 38985 | 38985 | 38938 | **35060** |
| DR7 | 53645 | 54645 | 54645 | 54607 | **52153** |
| DR8 | 2904060 | 2905060 | 2905056 | 2905060 | **2340639** |
| DR9 | 349000 | 350000 | 349967 | 350000 | **303402** |
| DR10 | 324980 | 325980 | 325979 | 325978 | **274405** |
| DR11 | 1628127 | 1629168 | 1629103 | 1629170 | **1297762** |
| DR12 | 2071276 | 2073230 | 2072653 | 2074116 | **1816915** |
| DR13 | 1199000 | 1200000 | 1199959 | 1200000 | **1032475** |

Eigen-decomposition time and the lesser the memory requirement. Table 6 shows the size of the bipartite graphs used by different SC methods, which indicates that the proposed graph carries fewer edges than the other graphs.

### 5.4 Ablation study

A couple of parameters, e.g., $k'$ and the number of rounds of MST $k$ influence SBSC's performance. $k'$ is a parameter of the partitioning technique that affects the quality of representatives, and the number of rounds $k$ decides the degree centrality of the graph. Fig. 8 shows the CA and execution time of SBSC on four datasets (DS2, DS4, DR6, and, DR10) by varying the value of $k'$ from 4 to 15. As discussed in Table 2, DS2 carries non-convex-shaped clusters, DS4 is large-sized, DS6 is high-dimensional, and DS10 possesses many clusters. Fig. 8 indicates that either the highest CA score is achieved or there is no substantial change in the CA score for $k' \geq 10$ in all the cases. Also, the execution time significantly increases for $k' \geq 10$ in all the four datasets. Therefore, the value of $k'$ is set to 10.

Similarly, Fig. 9 plots the CA and execution time of SBSC by varying the value of $k$ from 2 to 5. As can be observed from Fig. 9, for all four datasets, either the highest CA is achieved or there is not much substantial change in the CA score for $k \geq 2$. However, the execution time substantially increases with the increase in the rounds of MST $k$. Therefore, we set the value of $k$ as 2.

## 6 Conclusion and future work

We proposed a fast spectral clustering method (SBSC) that builds a sparse bipartite proximity graph using the MST of sub-linear representatives to discover better clusters without tuning any extrinsic variables. SBSC first identified the effective representative points by incorporating the border points-based partitioning, then determined the neighbors of each representative point using the MST to gather the local information of instances efficiently. The proposed proximity graph is highly sparse, i.e., of linear size, which contributes towards improving the computational time of projecting the data into the Eigen-space. Experimental analyses on diversified large datasets

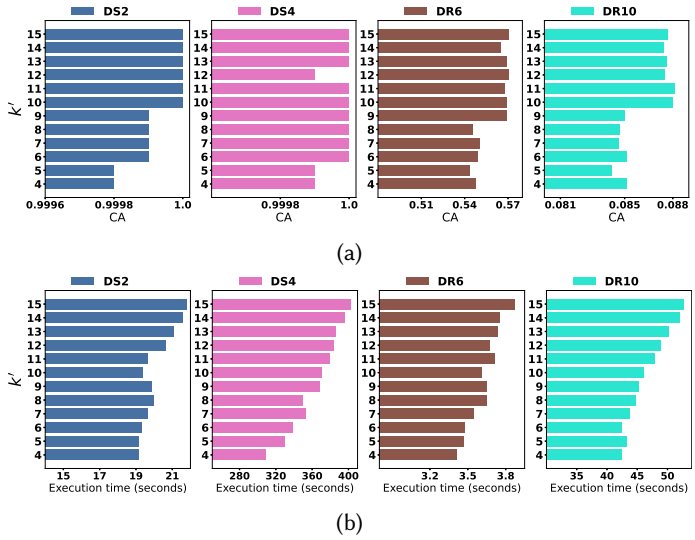



Fig. 8. Clustering accuracy (CA) and execution time of SBSC by varying the value of $k'$ variable: (a) CA. (b) Execution time in seconds.

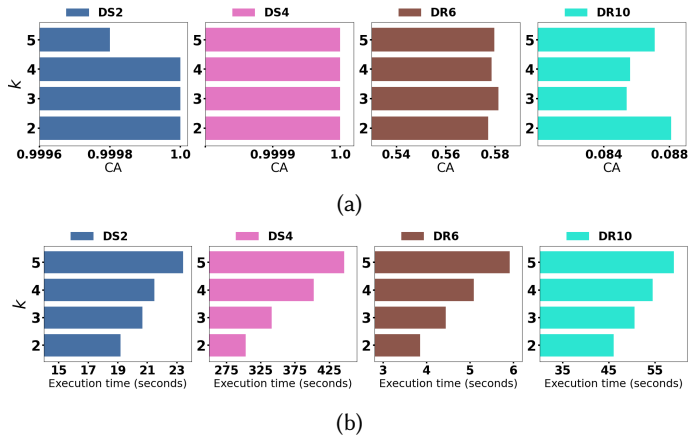



Fig. 9. CA and execution time of SBSC by varying the number of rounds ($k$) of MST: (a) CA. (b) Execution time in seconds.

suggested that SBSC detects more suitable clusters than other competing algorithms with the least execution time.

The future scope is to employ the proposed proximity graph in other suitable unsupervised algorithms like density-based methods for better clustering results. Another future work is to make BSC methods more efficient by introducing parallelism to its steps, such as applying the parallel $K$-means algorithm [15, 38] to compute representatives and incorporating parallel Eigensolvers [3, 38] in the Eigen-decomposition step. Moreover, some steps of SBSC can be parallelized, e.g., the second stage partitioning of SBSC produces partitions by applying the $K$-means algorithm

separately on every partition of the first stage. Additionally, the computation of two-hops neighbors of representatives could also be parallelized.

## References


[1] Amit Aflalo, Shai Bagon, Tamar Kashti, and Yonina Eldar. 2023. DeepCut: Unsupervised segmentation using graph neural networks clustering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*. 32–41.

[2] Deng Cai and Xinlei Chen. 2015. Large Scale Spectral Clustering Via Landmark-Based Sparse Representation. *IEEE Transactions on Cybernetics* 45, 8 (2015), 1669–1680. doi:10.1109/TCYB.2014.2358564

[3] Wen-Yen Chen, Yangqiu Song, Hongjie Bai, Chih-Jen Lin, and Edward Y. Chang. 2011. Parallel Spectral Clustering in Distributed Systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 3 (2011), 568–586. doi:10.1109/TPAMI.2010.88

[4] Dongdong Cheng, Jinlong Huang, Sulan Zhang, Xiaohua Zhang, and Xin Luo. 2022. A Novel Approximate Spectral Clustering Algorithm With Dense Cores and Density Peaks. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 52, 4 (2022), 2348–2360. doi:10.1109/TSMC.2021.3049490

[5] Anna Choromanska, Tony Jebara, Hyungtae Kim, Mahesh Mohan, and Claire Monteleoni. 2013. Fast Spectral Clustering via the Nyström Method. In *Algorithmic Learning Theory*, Sanjay Jain, Rémi Munos, Frank Stephan, and Thomas Zeugmann (Eds.). Springer, Berlin, Heidelberg, 367–381.

[6] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. 2017. EMNIST: Extending MNIST to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2921–2926. doi:10.1109/IJCNN.2017.7966217

[7] Dheeru Dua and Casey Graff. April, 2024. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[8] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise.. In *KDD*, Vol. 96. 226–231.

[9] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. 2004. Spectral grouping using the Nystrom method. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26, 2 (2004), 214–225. doi:10.1109/TPAMI.2004.1262185

[10] Gene H Golub and Charles F Van Loan. 2013. *Matrix computations*. Vol. 3. JHU press.

[11] Ellen Hohma, Christian M. M. Frey, Anna Beer, and Thomas Seidl. 2022. SCAR: spectral clustering accelerated and robustified. *Proceedings of the VLDB Endowment* 15, 11 (July 2022), 3031–3044. doi:10.14778/3551793.3551850

[12] Xia Hong, Junbin Gao, Hong Wei, James Xiao, and Richard Mitchell. 2023. Two-step scalable spectral clustering algorithm using landmarks and probability density estimation. *Neurocomputing* 519 (2023), 173–186.

[13] Dong Huang, Chang-Dong Wang, Jian-Sheng Wu, Jian-Huang Lai, and Chee-Keong Kwoh. 2019. Ultra-scalable spectral clustering and ensemble clustering. *IEEE Transactions on Knowledge and Data Engineering* 32, 6 (2019), 1212–1226.

[14] Qiang Huang, Pingyi Luo, and Anthony K. H. Tung. 2023. A New Sparse Data Clustering Method Based On Frequent Items. *Proceedings of the ACM on Management of Data* 1, 1, Article 5 (may 2023), 28 pages. doi:10.1145/3588685

[15] Zenan Huo, Gang Mei, Giampaolo Casolla, and Fabio Giampaolo. 2020. Designing an efficient parallel spectral clustering algorithm on multi-core processors in Julia. *J. Parallel and Distrib. Comput.* 138 (2020), 211–221. doi:10.1016/j.jpdc.2020.01.003

[16] Anil K Jain. 2010. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters* 31, 8 (2010), 651–666.

[17] R Janani and S Vijayarani. 2019. Text document clustering using spectral clustering algorithm with particle swarm optimization. *Expert Systems with Applications* 134 (2019), 192–200.

[18] Stephen C Johnson. 1967. Hierarchical clustering schemes. *Psychometrika* 32, 3 (1967), 241–254.

[19] R Jothi, Sraban Kumar Mohanty, and Aparajita Ojha. 2018. Fast approximate minimum spanning tree based clustering algorithm. *Neurocomputing* 272 (2018), 542–557.

[20] Abdul Atif Khan. March, 2025. SBSC Source Code. https://github.com/k-atif/SBSC

[21] Abdul Atif Khan, Mohammad Maksood Akhter, Rashmi Maheshwari, and Sraban Kumar Mohanty. 2024. L-ASCRA: A Linearithmic Time Approximate Spectral Clustering Algorithm Using Topologically-Preserved Representatives. *IEEE Transactions on Knowledge and Data Engineering* 36, 12 (2024), 8643–8654. doi:10.1109/TKDE.2024.3483572

[22] Abdul Atif Khan and Sraban Kumar Mohanty. 2022. A fast spectral clustering technique using MST based proximity graph for diversified datasets. *Information Sciences* 609 (2022), 1113–1131.

[23] Hongmin Li, Xiucai Ye, Akira Imakura, and Tetsuya Sakurai. 2022. Divide-and-conquer based large-scale spectral clustering. *Neurocomputing* 501 (2022), 664–678. doi:10.1016/j.neucom.2022.06.006

[24] Xiang Li, Ben Kao, Caihua Shan, Dawei Yin, and Martin Ester. 2020. CAST: A Correlation-based Adaptive Spectral Clustering Algorithm on Multi-scale Data. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Virtual Event, CA, USA) *(KDD '20)*. Association for Computing Machinery, New York, NY, USA, 439–449. doi:10.1145/3394486.3403086

[25] Yunfan Li, Peng Hu, Zitao Liu, Dezhong Peng, Joey Tianyi Zhou, and Xi Peng. 2021. Contrastive Clustering. *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 10 (May 2021), 8547–8555. doi:10.1609/aaai.v35i10.17037

[26] Zhenguo Li, Xiao-Ming Wu, and Shih-Fu Chang. 2012. Segmentation using superpixels: A bipartite graph partitioning approach. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*. 789–796. doi:10.1109/CVPR.2012.6247750

[27] James MacQueen et al. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, Vol. 1. Oakland, CA, USA, 281–297.

[28] Sridhar Mahadevan. 2008. Fast Spectral Learning using Lanczos Eigenspace Projections. In *AAAI*. 1472–1475.

[29] Bojan Mohar, Y Alavi, G Chartrand, and OR Oellermann. 1991. The Laplacian spectrum of graphs. *Graph theory, Combinatorics, and Applications* 2, 871-898 (1991), 12.

[30] Andrew Ng, Michael Jordan, and Yair Weiss. 2001. On Spectral Clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, Vol. 14. MIT Press. https://proceedings.neurips.cc/paper_files/paper/2001/file/801272ee79cfde7fa5960571fee36b9b-Paper.pdf

[31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[32] Julien Perolat, Inés Couso, Kevin Loquin, and Olivier Strauss. 2015. Generalizing the Wilcoxon rank-sum test for interval data. *International Journal of Approximate Reasoning* 56 (2015), 108–121. doi:10.1016/j.ijar.2014.08.001

[33] Teng Qiu and Yong-Jie Li. 2023. Fast LDP-MST: An Efficient Density-Peak-Based Clustering Method for Large-Size Datasets. *IEEE Transactions on Knowledge and Data Engineering* 35, 5 (2023), 4767–4780. doi:10.1109/TKDE.2022.3150403

[34] William M Rand. 1971. Objective criteria for the evaluation of clustering methods. *J. Amer. Statist. Assoc.* 66, 336 (1971), 846–850. doi:10.1080/01621459.1971.10482356

[35] Lina Ren, Ruizhang Huang, Shengwei Ma, Yongbin Qin, Yanping Chen, and Chuan Lin. 2023. Deep Multi-kernel Clustering Network. In *2023 IEEE International Conference on Data Mining (ICDM)*. 528–537. doi:10.1109/ICDM58522.2023.00062

[36] Yazhou Ren, Ni Wang, Mingxia Li, and Zenglin Xu. 2020. Deep density-based image clustering. *Knowledge-Based Systems* 197 (2020), 105841.

[37] Alex Rodriguez and Alessandro Laio. 2014. Clustering by fast search and find of density peaks. *Science* 344, 6191 (2014), 1492–1496. doi:10.1126/science.1242072

[38] Krishna Kumar Sharma and Ayan Seal. 2021. Spectral embedded generalized mean based k-nearest neighbors clustering with S-distance. *Expert Systems with Applications* 169 (2021), 114326.

[39] Jianbo Shi and J. Malik. 2000. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22, 8 (2000), 888–905. doi:10.1109/34.868688

[40] Kadim Taşdemir, Berna Yalçin, and Isa Yildirim. 2015. Approximate spectral clustering with utilized similarity information using geodesic based hybrid distance measures. *Pattern Recognition* 48, 4 (2015), 1465–1477.

[41] Ulrike Von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and Computing* 17, 4 (2007), 395–416.

[42] Lingfei Wu, Pin-Yu Chen, Ian En-Hsu Yen, Fangli Xu, Yinglong Xia, and Charu Aggarwal. 2018. Scalable Spectral Clustering Using Random Binning Features. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (London, United Kingdom) *(KDD '18)*. Association for Computing Machinery, New York, NY, USA, 2506–2515. doi:10.1145/3219819.3220090

[43] Jiang Xie, Weiyu Kong, Shuyin Xia, Guoyin Wang, and Xinbo Gao. 2023. An Efficient Spectral Clustering Algorithm Based on Granular-Ball. *IEEE Transactions on Knowledge and Data Engineering* 35, 9 (2023), 9743–9753. doi:10.1109/TKDE.2023.3249475

[44] Wei Xu, Xin Liu, and Yihong Gong. 2003. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval* (Toronto, Canada) *(SIGIR '03)*. Association for Computing Machinery, New York, NY, USA, 267–273. doi:10.1145/860435.860485

[45] Donghui Yan, Ling Huang, and Michael I. Jordan. 2009. Fast approximate spectral clustering. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Paris, France) *(KDD '09)*. Association for Computing Machinery, New York, NY, USA, 907–916. doi:10.1145/1557019.1557118

[46] Geping Yang, Sucheng Deng, Can Chen, Yiyang Yang, Zhiguo Gong, Xiang Chen, and Zhifeng Hao. 2023. LiteWSEC: A Lightweight Framework for Web-Scale Spectral Ensemble Clustering. *IEEE Transactions on Knowledge and Data Engineering* 35, 10 (2023), 10035–10047. doi:10.1109/TKDE.2023.3267167

[47] Caiming Zhong, Mikko Malinen, Duoqian Miao, and Pasi Fränti. 2015. A fast minimum spanning tree algorithm based on K-means. *Information Sciences* 295 (2015), 1–17.

Received October 2024; revised January 2025; accepted February 2025