Graph Laplacian for Heterogeneous Data Clustering in Sensor-Based Internet of Things

Vishal Krishna Singh, Gaurav Tripathi, Aman Ojha, Rajat Bhardwaj, and Haider Raza

V. K. Singh is the Head of Wireless Communications and Analytics Research Lab at Indian Institute of Information Technology, Lucknow, UP 226002 India (corresponding author to provide phone: 639-481-7065; e-mail: vashukrishna@gmail.com).

G. Tripathi is with Wireless Communications and Analytics Research Lab at Indian Institute of Information Technology, Lucknow, UP 226002 India (e-mail: gt.mnnit@gmail.com).

A. Ojha was with Wireless Communications and Analytics Research Lab at Indian Institute of Information Technology, Lucknow, UP 226002 India. He is now with the Intel Corporation, Bengaluru, KA 560103 India (e-mail: amanojha33@gmail.com).

R. Bhardwaj is with H&M AI, Lucknow, UP 226002 India (e-mail: arjun67d@gmail.com).

H. Raza is with University of Essex, Wivenhoe Park, Colchester CO4 3SQ India (e-mail: h.raza@essex.ac.uk).

ABSTRACT

Traditional clustering algorithms are not suited for heterogeneous data of Sensor-Based Internet of Things. The accuracy of real-time data processing, in such applications, is further compromised because of the noise and missing values in the data. Considering the need for accurate clustering, a Graph Laplacian based heterogeneous data clustering is proposed in this work. Exploiting the correlation structure of the data, weight graphs are used to generate a Graph Laplacian matrix to obtain co-related data points. Eigen values are further used to obtain distance based accurate clusters. The proposed algorithm is validated on five different real-world data sets and is able to outperform most of the existing algorithms. A detailed mathematical analysis followed by extensive simulation on real-world data sets, proves the dexterity of the proposed method as the performance gap, with respect to the state-of-the-art methods, in terms of accuracy and purity is as high as 30%.

Keywords:

Spectral Clustering, Heterogeneous Data, Machine Learning, Internet of Things, Sensor-based IoT, heterogeneous data.

1. INTRODUCTION

A Sensor-Based Internet of Things (SBIoT) ecosystem is composed of hardware devices like sensors, micro-controllers, and communication hardware such as gateways and CPUs to transmit data on a web-enabled platform for real-time processing. Often, the data obtained is not homogeneous and is corrupted by noise and redundancy. The real-time processing of such data necessitates the use of Machine Learning (ML) and Artificial Intelligence (AI) for reliable data collection, transmission, processing, and classification in various SBIoT applications. The use of ML in SBIoT allows insights, previously buried in data, allowing for faster and better decision-making. However, in a SBIoT framework, the data often received is not only heterogeneous but also hugely sparse with null values. Considering the constraints, imposed due to the heterogeneity of the SBIoT data, traditional clustering algorithms are ineffective as homogeneous and clean dataset is required without any missing values [16], [17], [18] and [19].

Existing research presents many interesting ideas to deal with such issues like dimensionality reduction, filling of missing values, interpolation, or approximation. These algorithms address the issues of large data sets, missing value features, and irregular data shapes. However, these methods are limited to dealing with numerical characteristics. Other techniques, such as those suggested by [1], [2], [3], [4] and [5], are not designed to handle data ambiguity, which is a common issue in many real-world applications. The authors in [6] and [7] proposed two techniques that can handle both uncertainty and heterogeneous data, by developing techniques on rough sets. Other methods include, Adaptive Weights Clustering (AWC) [8], which is based on the idea of locally weighting each point (document, abstract) in terms of cluster membership, however, accuracy and clustering purity remain open issues for further research. The authors in [9] propose a method to cluster the dataset using graph Laplacian by first approximating the graph and then integrating the Laplacians for preserving all the cluster information in multiple graphs. A method proposed in [10] uses a matrix completion algorithm for fast similarity matrix calculation. Although it retains the accuracy, but the performance may further be improved by optimizing the robustness of the proposed approach. A novel approach of mixed order spectral clustering framework is proposed in [11] to synchronously model second and third order structures both. The limitation of this framework is that it cannot mix more than two orders. Multi-view spectral clustering shows notable performance while capturing correlations, but it is inefficient and not suitable for large datasets. To address this issue [12] proposes a multi-view spectral clustering model, that shows significantly improved results than the existing methods.

Thus, to overcome the drawbacks of existing approaches on

heterogeneous and noisy data sets, a novel graph Laplacian based clustering algorithm is proposed in this work. The main contributions of this paper are as follows:

1. A novel graph Laplacian based clustering algorithm for heterogeneous and noisy data of SBIoT.
2. A detailed analytical analysis of the proposed method.
3. An extensive simulation study on 5 real world data sets along with a detailed comparative analysis of the results.

2. PROBLEM DESCRIPTION

In an SBIoT framework, the data often received is not homogeneous but hugely sparse and heterogeneous with null values. Traditional clustering algorithms require homogeneous and clean datasets without any missing values. Moreover, the existing methods fail to overcome the challenges imposed due to sparse data matrices and often suffer from data loss. Dropping the null data, in such cases, results in almost empty data matrices. On the other hand, speculating the missing values, results in steep degradation in clustering accuracy. As such, the existing methods fail to deliver accurate classification of clustered data in case of sparse or small data sets. The issues associated with clustering heterogeneous data in SBIoT applications are thus summarized as:

1) Data, in the SBIoT framework, is often heterogeneous and hugely sparse with null values. Moreover, cleaning data like dropping data points with missing values often leads to a huge loss of data.
2) Dimensionality Reduction and missing value speculation often lead to low accuracy and high running cost.
3) The existing clustering methods suffer from the inability to identify the non-convex patterns in the heterogeneous data because of which, adversely impacts the clustering accuracy.

3. SYSTEM MODEL

An SBIoT network with heterogeneous observations is realized for a smart park, where simultaneous events are observed and reported for real-time processing by randomly deployed nodes. Although, many different types of events are sensed and reported only five different observations are considered to be reported by every node, viz. temperature, humidity, pressure, light, and air quality. Events are reported as and when they occur via the deployed network of nodes, which are otherwise programmed to conserve energy. To realize a real-world scenario, the smart park is assumed to have rough weather conditions (i.e., frequent rain and wind) where data loss, link loss, and node loss are frequent. The smart park is equipped with smart lights and smart parking areas in addition to dense trees, smart cycle track, and a pool as shown in Figure 1. The symbols used in this work are described in Table 1.

4. DATASET DESCRIPTION & PRE-PROCESSING

The performance of the proposed algorithm is tested on five datasets, namely WINE, E.Coli, Abalone, HTRU2, and Stone datasets. All of these data sets are obtained from the open-source UCI repository forum. Table VII shows the various datasets used with a brief description. To meet the needs of the proposed work, the data sets are introduced to a prepossessing step which involves the introduction of different ratios of null values to get different heterogeneous data sets. Precisely, a maximum of 20% null values, in a complete data set, are introduced for validating the proposed algorithm.

Table 1: Symbols used

Symbol	Description	Symbol	Description
l_1	Manhattan Distance $(\|y_2 - y_1\| + \|x_2 - x_1\|)$	D	Degree matrix
l_2	Euclidean Distance $\left(\sqrt{(\|y_2 - y_1\|^2 + \|x_2 - x_1\|^2)}\right)$	A	Adjacency matrix
G	Graph of size n	L	Laplacian matrix
x	Number of features	ACC	Accuracy Score
c	Threshold weight	d_i	degree of the i^{th} node
W	Weight matrix of size $n \times n$	SIL	Silhouette Score
N	Number of objects (data points)	k	Number of clusters
t_j	Classification which has the max count for cluster c_i	c_i	Cluster in C
a	Average intra-cluster distance	b	Average inter-cluster distance

Table 2: UCI Real-world Dataset Description

	Samples	Features	Clusters
Wine	178	13	3
E-coli	336	7	8
Abalone	4177	8	28
HTRU2	17898	8	2
Stone	79	8	4

5. PROPOSED METHODOLOGY

The proposed scheme is based on the general idea that the related observations, from an SBIoT device, exhibit a strong correlation structure. As such, a weight graph is constructed to get a relative distance between the participating nodes which is used for creating a graph with edges of weight as the same relative distance. This is followed by the construction of a graph Laplacian matrix using degree matrix and adjacency matrix. Finally, the graph Laplacian matrix is used to obtain the desired clusters based on the eigenvalues and corresponding eigenvectors.

5.1 Construction of Weight Graph

The stepwise construction of the weight graph is explained below:

- Create a new matrix of size n × n where n is the total number of rows in the data set. The default value is set to ∞ and that will be the final weight matrix in end.

- Start from the first row, and for every data point or row, iterate on all other data points ahead of it and get an intersection of features of those two rows, and drop columns of both rows not in the intersection.
- Compute all the relative distances and choose the one with optimal accuracy.

The steps of the algorithm are shown in Algorithm 1. Since, Algorithm 1, has 2 nested loops, the time complexity is calculated as the number of times the innermost loop is executed. As evident, the Inner loop runs 1 less number of times the value of the outer loop, thus:

$$\mathbf{1} + \mathbf{2} + \cdots + (\boldsymbol{n} - \mathbf{2}) + (\boldsymbol{n} - \mathbf{1}) = \frac{(\boldsymbol{n}-\mathbf{1})(\boldsymbol{n})}{\mathbf{2}} = \boldsymbol{O}(\boldsymbol{n}^{\mathbf{2}}) \quad \textbf{(1)}$$

Algorithm 1 Weight Graph

1. **for** *iteration* = 1,2,...,*N* do
2. **for** actor = iteration +1…N do
3. Take inner Join/Intersection of both data points as Z.
4. **if** Z==0 then
5. Since no common variables, Go To line 2.
6. **end if**
7. Compute pairwise distance or any optimal relative distance and store at position (iteration, actor)
8. **end for**
9. **end for**

Consider the case where six sensor nodes with the five predefined feature sets, are deployed for reporting the events. A sample node data set and the weight graph *W* for an $N \times N$ matrix (N = number of samples = 6 in this case), are shown in Table 2 and Table 3 respectively.

Table 3: Sensor Nodes Data Set

A	B	C	D	E
-	-	C1	-	-
A2	B2	C2	-	-
A3	-	C3	-	-
-	B4	-	D4	E4
A5	B5	-	-	-
A6	B6	C6	D6	-

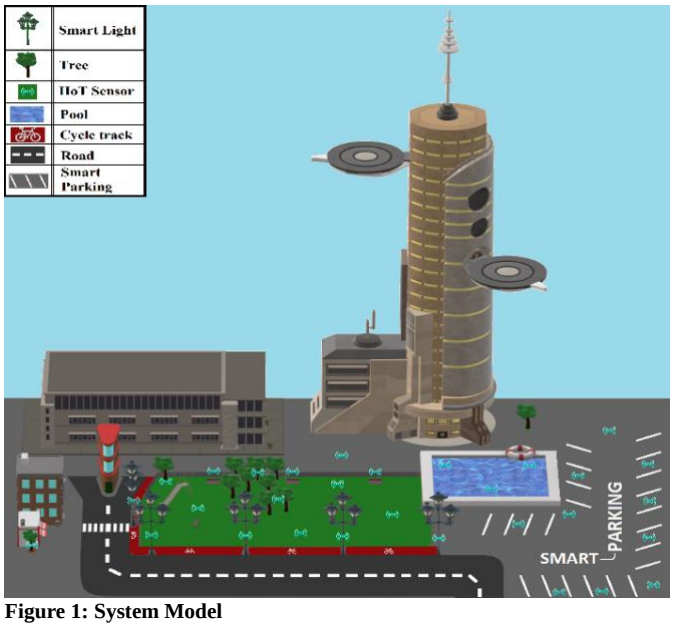


Figure 1: System Model

Table 4: Weight Graph

	1	2	3	4	5	6
1	∞	∞	8	∞	∞	8
2	∞	∞	8	1	6	8
3	8	8	∞	∞	1	3
4	∞	1	∞	∞	2	5
5	∞	6	1	2	∞	6
6	8	8	3	5	6	∞

A threshold weight *c*, chosen based on the optimized final accuracy, is used for the construction of a similarity graph. The entries up to *c* in the weight graph are marked as 1 and those above *c*, are marked as 0. The similarity graph, thus obtained for the weight graph *W*, is shown in Table 4.

Table 5: Similarity Graph (S): c = 5

	1	2	3	4	5	6
1	0	0	1	0	0	1
2	0	0	1	0	1	1
3	1	1	0	0	0	0
4	0	0	0	0	0	1
5	0	1	0	0	0	1
6	1	1	0	1	1	0

5.2 Construction of Graph Laplacian Matrix

The steps for the construction of graph Laplacian matrix, are as follows:

- Create an adjacency matrix (*A*) using the similarity graph created in subsection 4.1.
- Create a degree matrix (*D*) using the same similarity graph created in subsection 4.1 using equation 2 and 3.
- Get the graph Laplacian matrix (L) using degree and adjacency matrix by following equation 4.

The adjacency matrix (A) is created using the similarity graph created in subsection 4.1. The degree matrix (D) is then created by adding all the points of the specific row. It is empirical to note that the cells are filled along the diagonal for each row of the degree matrix. The points of the degree matrix may be extracted by the degree of the corresponding node. The degree of the i^{th} node is given by:

$$d_i = \sum_{j=1|(i,j)\epsilon E}^{n} w_{ij} \quad \textbf{(2)}$$

where w_{ij} is the edge/weight between the nodes i and j as mentioned in the adjacency matrix (A). Thus, the degree matrix is defined as:

$$\boldsymbol{D_{ij}} = \begin{cases} \boldsymbol{d_i}, \boldsymbol{i = j} \\ \mathbf{0}, \boldsymbol{i \neq j} \end{cases} \quad \textbf{(3)}$$

The Table 5 shows the degree matrix obtained for the example, as explained in subsection 4.1. To obtain the normal Laplacian, the adjacency matrix is subtracted from the degree matrix (as shown in equation 4). The Table 6 shows the Graph Laplacian matrix (L) for the discussed example. The degree of every node is shown by the Laplacian's diagonal, while the negative edge weights are represented by the off diagonal.

$$\boldsymbol{L = D - A} \quad \textbf{(4)}$$

Table 6: Degree Matrix (D)

	1	2	3	4	5	6
1	2	0	0	0	0	0
2	0	3	0	0	0	0
3	0	0	2	0	0	0
4	0	0	0	1	0	0
5	0	0	0	0	2	0
6	0	0	0	0	0	4

Table 7: Graph Laplacian Matrix (L)

	1	2	3	4	5	6
1	2	0	-1	0	0	-1
2	0	3	-1	0	-1	-1
3	-1	-1	2	0	0	0
4	0	0	0	1	0	-1
5	0	-1	0	0	2	-1
6	-1	-1	0	-1	-1	4

5.3 Compute Clusters using Eigenvalues

Once the Laplacian matrix (L) is obtained, the proposed algorithm takes advantage of one of its special properties to classify the data. Consider, if the graph (G) has *K* connected components, then *L* has *K* eigenvectors with an eigenvalue of 0. The steps are as follows:

- Calculate eigenvalues and eigenvectors of graph Laplacian matrix obtained from the subsection 4.2.
- Sort the eigenvalues and get *k* non-zero positive eigenvalues close to 0, where k is equal to the total number of clusters desired.
- Select the eigenvector with different eigenvalues only for best results. If case of same eigenvalues, it can be dropped.
- Stack those eigenvectors together to be clustered.
- Use *k*–means to classify the nodes based on their corresponding values in the eigenvector.

The corresponding eigenvalues and their respective eigenvectors are shown in the Figure 2. Since the example under consideration, has only one component, one eigenvalue is calculated as 0 out of the 6 eigenvalues. In a more complicated graph, there can be multiple components and thus multiple eigenvalues may be reported as 0. Figure 3a shows the plot of all the 6 eigenvalues, where the least three eigenvalues or the ones closest to 0, are selected and their respective eigenvectors are plotted. Figure 3b shows the plot of the first eigenvector with values closest to 0. Evidently, all the values are the same in this vector. Although there is only one eigenvalue with 0, Figure 3c shows the plot of the eigenvector of second and third eigenvalues which are closest to 0, and the second smallest eigenvalue.

```
eigenvalues:
[-0.000 0.882 1.451 2.534 3.865 5.269]

eigenvectors:
[[-0.408 -0.277  0.483 -0.652  0.024 -0.313]
 [-0.408 -0.181 -0.280  0.339  0.672 -0.395]
 [-0.408 -0.410  0.369  0.586 -0.373  0.216]
 [-0.408  0.841  0.231  0.155 -0.115 -0.189]
 [-0.408 -0.073 -0.699 -0.190 -0.536 -0.125]
 [-0.408  0.099 -0.104 -0.238  0.392  0.805]
```

Figure 2: Eigenvalues and Respective Eigenvector

5.4 Steps of the Algorithm

The goal is to construct a similarity graph with a data set such that a few of the data points are completely different and some are similar entries. Originally, the experiments included a concept to charge some cost on different dissimilar features. But multiple results proved not to go with the latter. Getting a number between every data point is somewhat like getting a distance between n-dimensional vector space where n is equal to the number of variables similar in both data points.
The following is a step-by-step description of the proposed algorithm:

Step 1: Construct a weight graph *W* using Algorithm 1.
Step 2: Define a threshold parameter *c* to create a similarity graph.
Step 3: Define a degree matrix *D* and adjacency matrix *A* from the similarity graph as described in subsection 4.2.
Step 4: Define a Laplacian matrix *L* as described in subsection 4.2.
Step 5: Find *k* largest eigenvectors and stack it in columns.
Step 6: Form the new matrix *Y* by re-normalizing each of *X*'s rows.
Step 7: Cluster each row of *Y* into *k* clusters using *k-means* or any other method, using each row as a point in R^k.
Step 8: If the row *i* in the matrix *Y* was allocated to cluster *j*, label the respective point s_i to cluster *j*.

6. ANALYTICAL ANALYSIS

There are many common designs for turning a collection of data points $x_1,...,x_n$ into a graph with pairwise similarity s_{ij} or pairwise distances d_{ij}. When creating similarity graphs, the aim is to represent the data points' local neighborhood connections. All the nodes with a distance less than the pre-defined threshold, as mentioned in subsection 4.1, are considered in this analysis. Additionally, the selected weights for this analysis do not give any additional information about that data set or for any graph. It is empirical to note that the directions are not considered and that all the edges are used to create undirected graph. Consider that, *G* is an undirected, weighted graph with a matrix *W*, and that $w_{ij} = w_{ji} \geq 0$.. It is not always assumed that eigenvectors of a matrix are normalized when utilizing them. The fixed vector ⊮ and a multiple a⊮ with some $a \neq 0$, are considered as the same eigenvectors. The eigenvalues are arranged in ascending order, with multiplicities taken into account. The eigenvectors corresponding to the *k* lowest eigenvalues are referred to as "the first *k* eigenvectors." The unnormalized graph Laplacian matrix (used in the proposed approach) is defined as in equation 4 i.e. $L = D - A$, where, the matrix *L* has the below properties:

For every vector $v \in R^n$ we have

$$v'Lv = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(v_i - v_j)^2 \qquad \textbf{(5)}$$

By the definition of d_i,

$$v'Lv = v'Dv - v'Wv = \sum_{i=1}^{n} d_i v_i^2 - \sum_{i,j=1}^{n} v_i v_j w_{ij}$$

$$= \frac{1}{2}\left(\sum_{i=1}^{n} d_i v_i^2 - 2\sum_{i,j=1}^{n} v_i v_j w_{ij} + \sum_{j=1}^{n} d_j v_j^2\right) \quad \textbf{(6)}$$

$$= \sum_{i,j=1}^{n} w_{ij}(v_i - v_j)^2 \quad \textbf{(7)}$$

- L is symmetric as it comes from the symmetry of D and weight matrix.
- L is positive and semi-definiteness as it is proved directly from the first property, which shows $v'Lv \geq 0$ for all $v \in R^n$.
- The lowest eigenvalue of the Laplacian matrix is zero.
- L has x positive, real-valued eigenvalues (with at least one 0) $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$.

Let G be the same graph, as mentioned above, with nonnegative weights. The multiplicity m of the eigenvalue zero of the Laplacian matrix is the same as the number of connected components $C_1,...,C_m$ in the graph. The eigen-space of the lowest eigenvalue is depicted by the same vectors $\mathbb{1}_{C_1},...,\mathbb{1}_{C_m}$ of those components.

Now consider the case where $m = 1$, which results in a connected graph. Assume that v is an eigenvector of lowest eigenvalue as zero, then

$$0 = v'Lv = \sum_{i,j=1}^{n} w_{ij}(v_i - v_j)^2 \quad \textbf{(8)}$$

Since the weights w_{ij} are always positive or zero, the equation 8 can only perish if all terms $w_{ij}(v_i - v_j)^2$ goes off. If two vertices v_i and v_j are joined (i.e., $w_{ij} > 0$), then v_i needs to be same as v_j. It is seen from this explanation that v must be constant for all nodes in the graph that may be joined by a path. Since all nodes of the same component and an un-directed network may be joined by a path, v must be fixed across the network. Now there is only a constant vector $\mathbb{1}$ as eigenvector of eigenvalue zero in a graph that is definitely the required vector of the joined component.

Considering an alternate situation when there are k linked components, it may be considered that the nodes are sorted as per to the joined components they hold to, without removing generality. Adjacency matrix A has a block diagonal shape in this case, with matrix L also being the same as:

$$L = \begin{pmatrix} L_1 & & & \\ & L_2 & & \\ & & \ddots & \\ & & & L_k \end{pmatrix} \quad \textbf{(9)}$$

It is worth noting that every part of L_i is itself a Laplacian graph matrix, as the Laplacian respective to the sub-graph of the i^{th} joined component.

The spectrum of Laplacian is given by the join of the spectra of L_i, as the respective eigenvectors of Laplacian are the eigenvectors of L_i containing zero at the locations of all the other parts which is the same for all parts of the diagonal matrix. Every L_i has eigenvalue zero, and the respective eigenvector is the fixed 1 vector on the i^{th} joined component, as each L_i is Laplacian of a joined graph. As a result, the matrix L has joined components the same as the number of eigenvalues equal to 0, and their respective eigenvectors are the joined component required vectors.

The key technique in the proposed method, described in section 4.3, is to alter the form of the abstract tuples a_i to points $b_i \in R^k$. This shift of representation is helpful because of the characteristics of the Laplacians matrix. This alters in representation improves the data's cluster characteristics, allowing clusters to be identified with ease in the new form.

The basic k -means clustering method, in particular, has no trouble detecting clusters in this new form.

7. SIMULATION SETUP

The system model, as explained in section III, is realized in Python for obtaining heterogeneous observations from 200 nodes, deployed to cover an ROI of 150m × 150m. A Unix based system with 4 GB 1600 MHz DDR4 memory and Macintosh operating system and 3.2 GHz Intel Core i5 processor is used to run the simulations. Additionally, jupyter notebook as IDE along with packages such as Pandas, Numpy, ScikitLearn (specifically k-means, spectral clustering to compare), Matplotlib, Scipy, etc. are used.

A comparative analysis of the proposed algorithm is presented for:

- Spectral clustering algorithm proposed in [13]: For the mentioned datasets, the proposed algorithm and the algorithm in [13], are implemented and tested on the same parameters.
- The algorithms proposed and used for comparative evaluations in [14] and [15]:
  For the mentioned datasets, the proposed algorithm and the algorithms proposed and used for comparative evaluations in [14] and [15], are implemented and tested on the same parameters.

The performance evaluation matrices, for comparative evaluation, are as follows:

- Purity is a unique parameter for evaluating cluster quality. It gives the percent of the total number of nodes or rows which were clustered correctly and is given by:

$$Purity = \frac{1}{N}\sum_{i=1}^{k} \max_j |c_i \cap t_j| \quad \textbf{(10)}$$

- Accuracy (ACC) is defined as the percentage of correct predictions for the test data and is given by:

$$accuracy = \frac{correct\_predictions}{all\_predictions} \quad \textbf{(11)}$$

The silhouette (SIL) value is a measure of how similar an object is to its cluster (cohesion) compared to other clusters (separation). The silhouette ranges from -1 to +1.

$$SilhouetteScore = \frac{(b-a)}{max(a,b)} \quad \textbf{(12)}$$

8. RESULTS AND COMPARATIVE ANALYSIS

8.1 Comparative Analysis with Algorithm in [13]

The proposed algorithm, along with the algorithm in [13] are implemented with common parameters and their performance on the five datasets is presented in Table 8. As evident from the results, the proposed algorithm is able to outperform the algorithm in [13] on almost all the datasets and metrices, except for purity in Wine and accuracy on Stone. The spectral algorithm in [13] could not be implemented for HTRU2 dataset, hence the results could not be compared. A comparative analysis shows that the proposed algorithm is able to achieve 1.13% better accuracy on wine dataset, 29.6% better accuracy on E.coli dataset, 1.17% better accuracy on Abalone dataset. A simple reason for the improved accuracy is the ability of the proposed method to identify the non-convex patterns in the heterogeneous data.

Table 8: Comparative Analysis with Algorithm in [13]

	Metric	Spectral [13]	Proposed Algorithm
Wine	Purity	99.43	98.87
	ACC	97.75	98.88
E-coli	Purity	60.41	68.45
	ACC	14.54	44.14
Abalone	Purity	23.29	27.45
	ACC	16.50	17.67
HTRU2	Purity	-	96.56
	ACC	-	96.89
Stone	Purity	97.26	98.20
	ACC	97.3	96.33

8.2 Comparative Analysis with Algorithm in [14] and [15]

Table 9 shows the complete comparison of the proposed algorithm to standard K-means, SSC, KKmeans, RSFKC, CLR, MEAP, k-MEAP, KMM as proposed and used in [14] and adacluster in [15]. A detailed analysis of the algorithm's performance on all the data sets is presented next.

8.3 Results on Wine Dataset

A careful observation of Table 9 reveals the fact that the proposed method is able to achieve up to 3% and 1% improvement in both purity and accuracy scores respectively compared to algorithms in [14] and [15]. It is empirical to note the proposed algorithm is able to deliver acceptable results even with null values while the other counterpart algorithms are unable to even execute with null variables. The comparative results with [13], [14], and [15] on purity and accuracy, shown in Figure 4c, clearly show the efficacy of the proposed method over existing methods.

Figure 4a shows the plot of eigenvalues of graph Laplacian of Wine dataset. It is evident from figure 4a that the values are not the same and only one eigenvalue is 0 which indicates clearly the only component in the graph. Evidently, the results obtained after k-means on the first 3 non-zero positive vectors, are better. The reason being that there are 3 clusters to be obtained for which the first 3 non-zero eigenvectors are used to cluster with different values. Figure 4b shows the plot of eigenvector of the 2^{nd}, 3^{rd}, and 4^{th} smallest eigenvalues. Since the number of clusters to be found is 3, only 3 eigenvectors are selected here apart from 1^{st} which is constant due to zero eigenvalue. As discussed in section 4, the more the spread of values, the better is the ability to divide the dataset into clusters. It can be clearly seen that dividing the dataset into 3 sections, using all the three vectors, gives better results.

8.4 Results on E.coli Dataset

Figure 5a shows the eigenvalues plot of graph Laplacian of E.coli dataset. Since there is only 1 component in the graph, the first eigenvalue is 0, but as shown in Figure 5a, all the other eigenvalues are the same as 336. Thus, selecting all the eigenvectors or only one won't affect the result. It is also important to note that although, the number of clusters desired is 8, but only one eigenvector will also give the same result as 8 eigenvectors when clustered. Figure 5b shows the plot of eigenvectors of second and third smallest eigenvalues as discussed in section 4, where it is shown that both eigenvectors will give the same results of dividing the data set. It can be seen that dividing the graph into 8 sections by 7 horizontal lines will remain the same, taking 1 or more eigenvectors. As evident from the observations reported in Table 9 and Figure 5c, the performance of the proposed algorithm is not at par with the other compared algorithms. The performance degradation is due to the same eigenvalues. However, as evident from Table 8, the proposed algorithm outperforms the standard spectral algorithm [13] implemented in scikit library. It showed up to 8% improvement in purity score and up to 30% improvement in accuracy score.

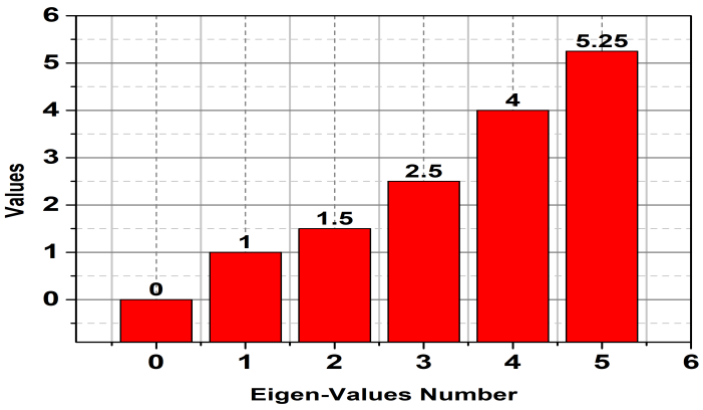
8.5 Results on Abalone Dataset

The performance comparison of the proposed approach, as presented in Table 9, proves that the proposed method is almost at par with the compared approaches in terms of purity. Moreover, the comparative results presented in Table 8 show the efficacy of the proposed algorithm over the standard spectral algorithm [13] implemented in scikit library. An improvement of 4% in purity score and up to 1% improvement in accuracy score evidently from the reported observations. Figure 6c presents the comparative results of the proposed method with the algorithms in [13], [14], and [15]. The eigenvalue plot (shown in Figure 6a) shows that there is only 1 component in the graph, thus the first eigenvalue is 0, but all the other eigenvalues are the same as 210. Thus, selecting all the eigenvectors or only 1, won't affect the results. Therefore, even though the number of clusters desired is 28, only one eigenvector will also give the same result as 28 eigenvectors when clustered.

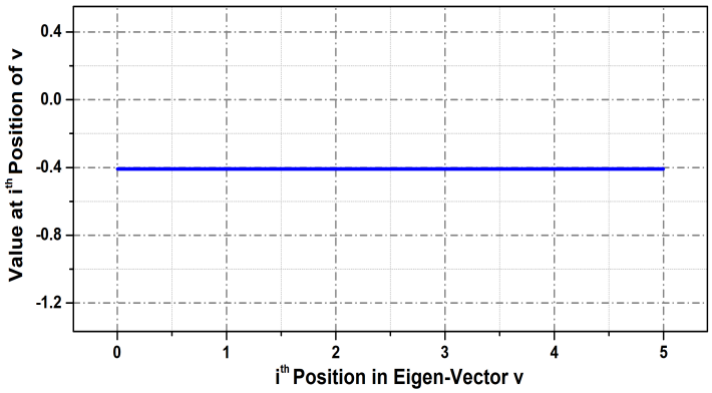
Figure 6b shows the plot of eigenvectors of the second, third, and fourth smallest eigenvalues. As discussed in Section 4, Figure 6b proves that all the eigenvectors give the same results of dividing the data set. It can be seen that dividing the graph into 28 sections by 27 horizontal lines will be the same, taking 1 or more eigenvectors. Evidently, due to the same eigenvalues, the reported accuracy and purity are at par with the other methods in [14]. This is one of the prime reasons for the improved performance of the proposed algorithm as compared to the standard spectral algorithm [13].

8.6 Results on HTRU2 Dataset

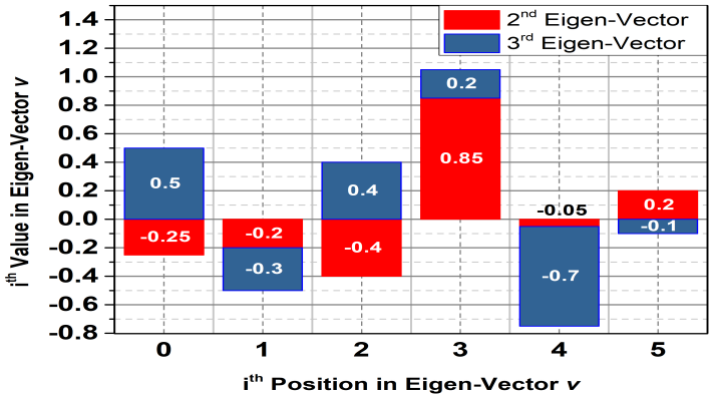
The comparative results, as presented in Table 9 and Figure 7c, show that the proposed algorithm is able to achieve significant improvement & shows up to 1.8% improvement in



(a) Total eigenvalues

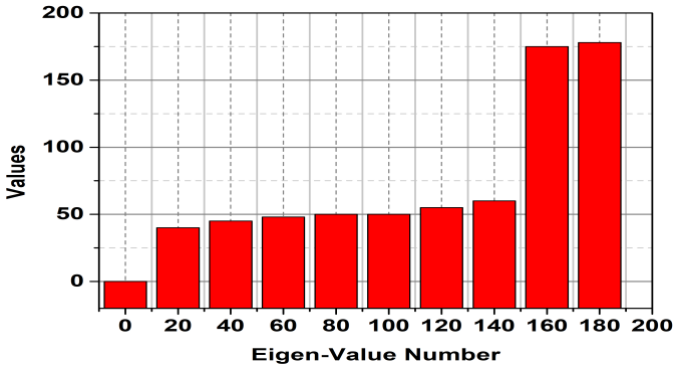


(b) Values in First eigenvector

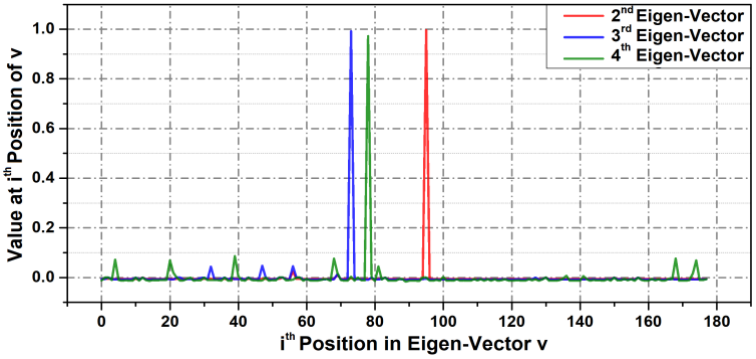


(c) Values in Second and Third eigenvector of eigenvalue Close to 0

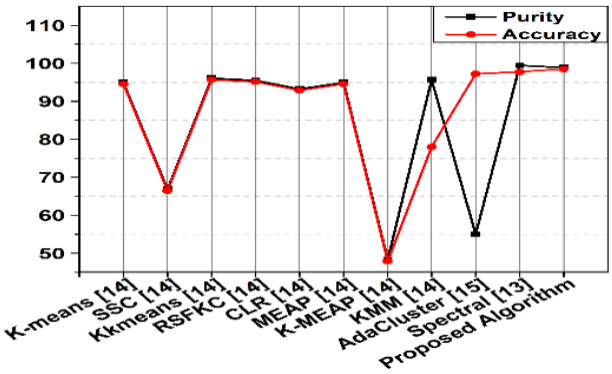
Fig 3. Eigenvalues and eigenvectors



(a) Eigenvalues plot of Wine dataset

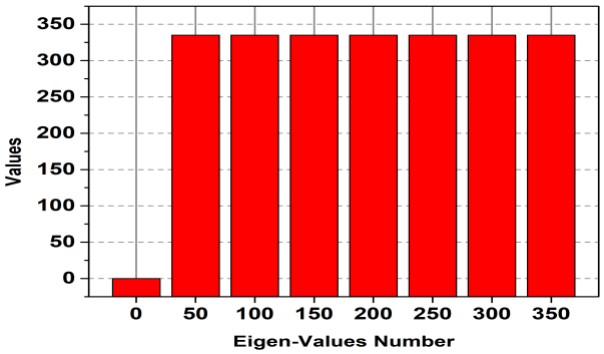


(b) Eigenvectors plot of Wine dataset

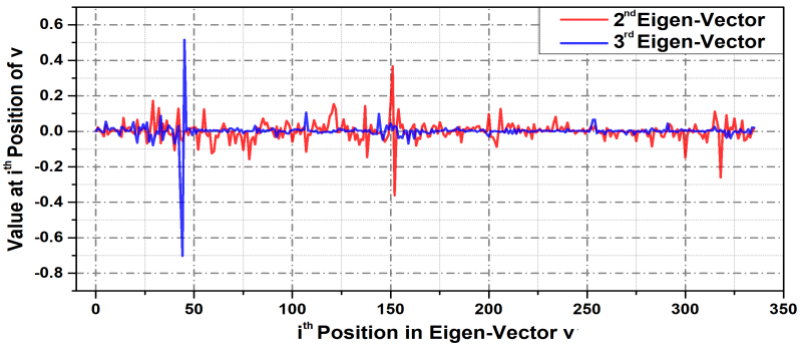


(c) Purity and Accuracy comparison of proposed algorithm on Wine dataset

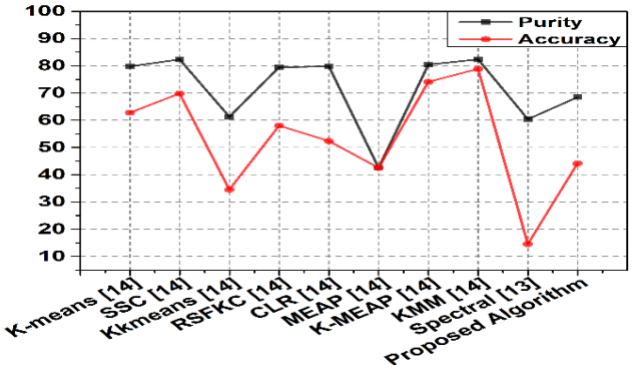
Fig. 4. Wine dataset



(a) Eigen values plot of E.coli dataset

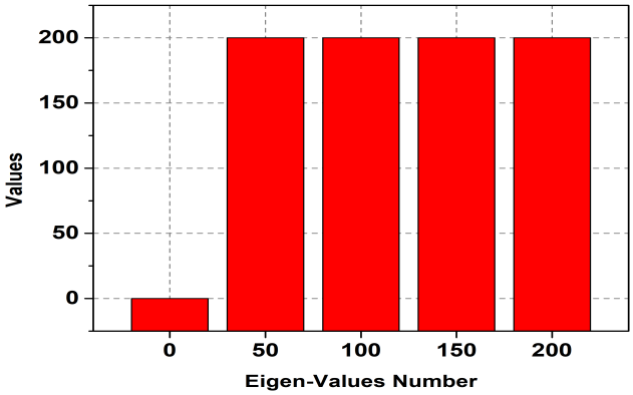


(b) Eigenvectors plot of E.coli dataset

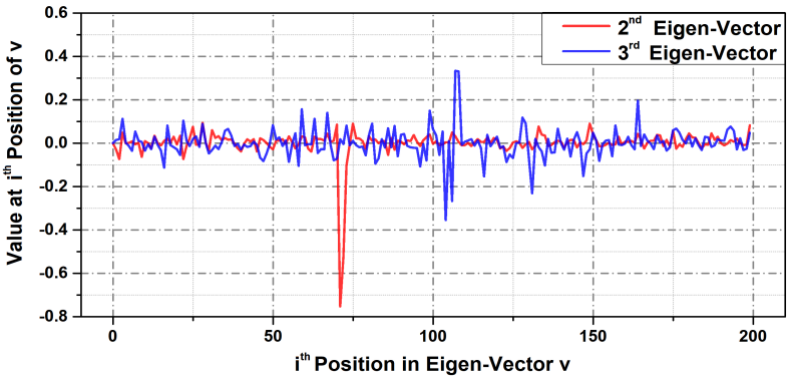


(c) Purity and Accuracy comparison of proposed algorithm on E.coli dataset

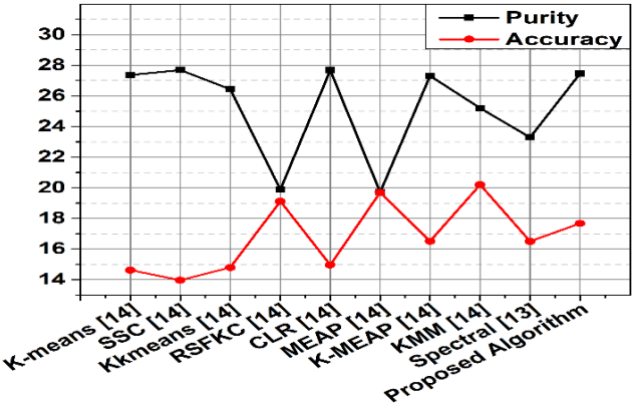
Fig. 5. E.coli dataset



(a) Eigenvalues plot of Abalone dataset

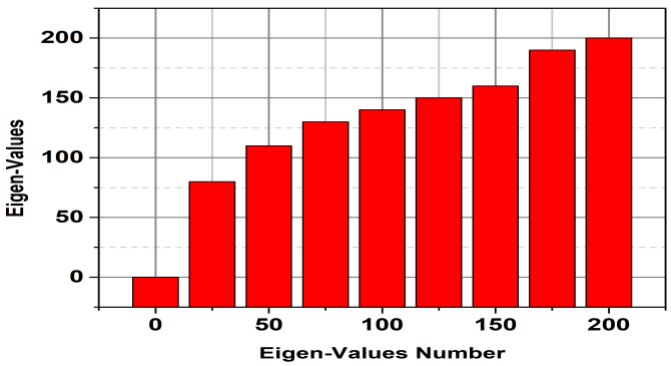


(b) Eigenvectors plot of Abalone dataset

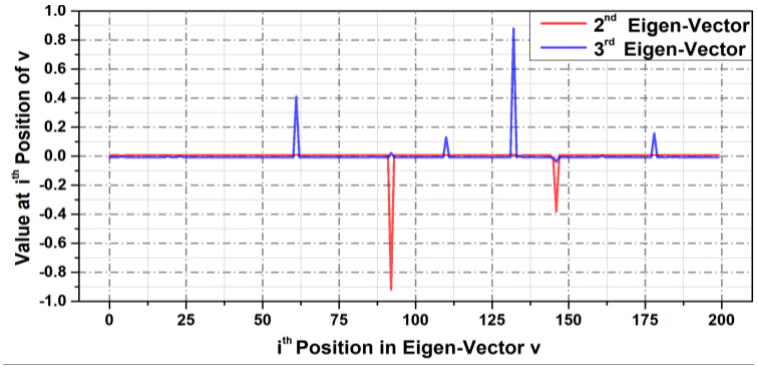


(c) Purity and Accuracy comparison of proposed algorithm on Abalone dataset

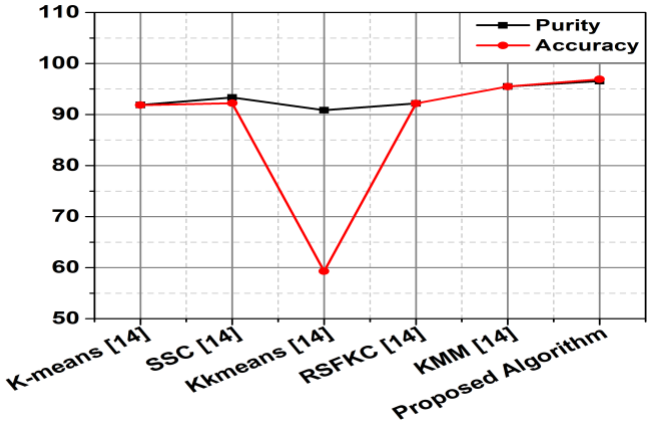
Fig. 6. Abalone dataset



(a) Eigenvalues plot of HTRU2 dataset

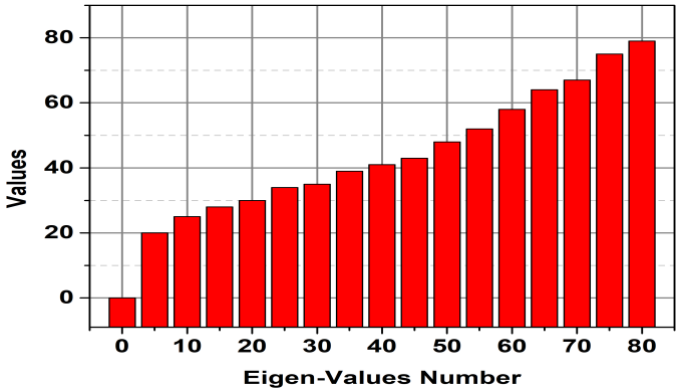


(b) Eigenvectors plot of HTRU2 dataset

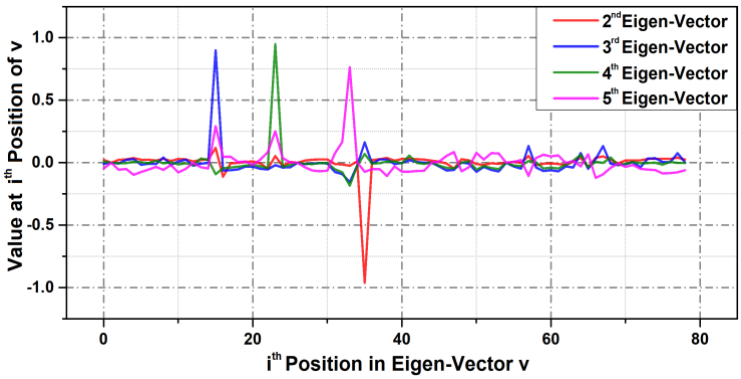


(c) Purity and Accuracy comparison of proposed algorithm on HTRU2 dataset

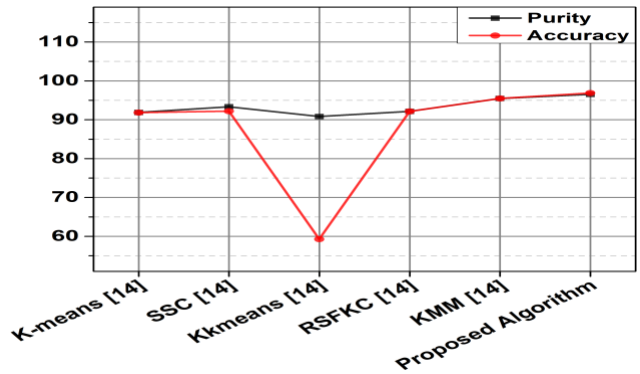
Fig. 7. HTRU2 dataset



(a) Eigen values plot of StoneFlakes dataset



(b) Eigenvectors plot of StoneFlakes dataset



(c) Purity and Accuracy comparison of proposed algorithm on StoneFlakes dataset

Fig. 8. StoneFlakes dataset

Table 9 Clustering Performance Comparison of Proposed Algorithm on Real-World Datasets (%)

	Mteric	K-means [14]	SSC [14]	KKmeans [14]	RSFKC [14]	CLR [14]	MEAP [14]	K-MEAP [14]	KMM [14]	AdaCluster [15]	Proposed Algorithm
Wine	Purity	94.94	66.85	96.06	95.50	93.25	94.94	48.31	95.76	-	98.87
	ACC	94.94	66.85	96.06	95.50	93.25	94.94	48.31	78	97.19	98.88
	SIL	-	-	-	-	-	-	-	-	-	43.71
E.coli	Purity	79.76	82.33	61.30	79.46	79.76	42.55	80.41	82.37	-	68.45
	ACC	62.79	59.82	34.52	58.03	52.38	42.55	74.10	78.85	-	44.14
	SIL	-	-	-	-	-	-	-	-	-	56.34
Abalone	Purity	27.36	27.68	26.43	19.89	27.67	19.70	27.31	25.20	-	27.45
	ACC	14.62	13.96	14.79	19.12	14.96	19.70	16.51	20.20	-	17.67
	SIL	-	-	-	-	-	-	-	-	-	00.57
HTRU2	Purity	91.89	93.35	90.84	92.17	-	-	-	95.49	-	96.56
	ACC	91.85	92.22	59.29	92.17	-	-	-	95.49	-	96.89
	SIL	-	-	-	-	-	-	-	-	-	41.36
StoneFlakes	Purity	-	-	-	-	-	-	-	-	-	98.20
	ACC	-	-	-	-	-	-	-	-	87.96	96.33
	SIL	-	-	-	-	-	-	-	-	-	47.36

both purity and accuracy scores compared to the KMM algorithm in [14] with the highest score. It is important to note that the proposed method gives a significantly, acceptable result even with null values, unlike the other algorithms which are unable to even execute under similar conditions. The plot of eigenvalues of graph Laplacian of HTRU2 dataset (figure 7a) and the plot of eigenvector of 2nd and 3rd smallest eigenvalue (figure 7b) proves the supremacy of the proposed method over existing approaches. Since the number of clusters to be found is 2, therefore, only 2 eigenvectors are selected, apart from 1st which is constant due to zero eigenvalue. As discussed in section 4, the more the spread of values, the better is the ability to divide the dataset into clusters, but it can be clearly seen that it's very hard to divide the dataset into 2 sections that will give the optimal results. Even in such adverse conditions, the proposed algorithm is able to deliver significantly better results as compared to all the other partition algorithms.

8.7 Results on StoneFlakes Dataset

The observations, as reported in Table 8 prove that the proposed method has a better purity score (about 1%) as compared to the standard spectral algorithm [13] implemented in scikit library. A simple reason for the marginally low accuracy of the proposed method is the high number of null values in the StoneFlakes dataset. When compared to the other methods, Table 9 and Figure 8b prove that the proposed algorithm is able to outperform most of its counterpart methods as the accuracy is as high as 8%. Figure 8a and 8b also confirm the efficiency of the proposed scheme over existing methods used for comparative analysis in [13] and [14].

9. Conclusion and Future Work

A novel method using the weight graph and graph Laplacian matrix for complete heterogeneous data set is proposed in this work. The improved accuracy and purity in clusters is because of the ability of the proposed method to consider non-convex patterns in the heterogeneous data and its ability to consider null values efficiently. Comparative analysis of the proposed method with state-of-the-art algorithms on 5 different real-world data sets proves the dexterity of the proposed algorithm. The performance improvement is shown in terms of accuracy, purity, and significantly better silhouette score (as high as 30% improvement) is reported based on the simulation analysis on the real-world data sets. We continue to work and test the proposed approach on large scale datasets and aim to present a detailed analysis in future.

References

1. Shukla, Ankita, Vishal Krishna Singh, and Mala Kalra. "Multimodal Device Clustering Using Mobile Agent for Correlation in Sensor-Based IoT." Adhoc & Sensor Wireless Networks 48 (2020).
2. A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," Journal of the Royal Statistical Society: Series B (Methodological), vol. 39, no. 1, pp. 1–22, 1977.
3. S. Guha, R. Rastogi, and K. Shim, "Rock: A robust clustering algorithm for categorical attributes," Information systems, vol. 25, no. 5, pp. 345– 366, 2000.
4. Z. Huang, "Extensions to the k-means algorithm for clustering large data sets with categorical values," Data mining and knowledge discovery, vol. 2, no. 3, pp. 283–304, 1998.
5. D. Gibson, J. Kleinberg, and P. Raghavan, "Clustering categorical data: An approach based on dynamical systems," The VLDB Journal, vol. 8, no. 3, pp. 222–236, 2000.
6. D. Parmar, T. Wu, and J. Blackhurst, "Mmr: an algorithm for clustering categorical data using rough set theory," Data & Knowledge Engineering, vol. 63, no. 3, pp. 879–893, 2007.
7. P. Kumar and B. Tripathy, "Mmer: an algorithm for clustering heterogeneous data using rough set theory," International Journal of Rapid Manufacturing, vol. 1, no. 2, pp. 189–207, 2009.
8. L. Adamyan, K. Efimov, C. Y. Chen, and W. K. Hardle, "Adaptive¨ weights clustering of research papers," Digital Finance, vol. 2, no. 3, pp. 169–187, 2020.
9. A. Khan and P. Maji, "Approximate Graph Laplacians for multimodal data clustering," IEEE Trans. on Pattern analysis and machine intelligence, vol. 43, no. 3, pp. 789–813, 2021.
10. X. Ma, S. Zhang, K. Pena-Pena, and G. R. Arce, "Fast spectral clustering method based on graph similarity matrix completion," Signal Processing, vol. 189, 2021.
11. Y. Ge, P. Peng, and H. Lu, "Mixed-order spectral clustering for complex networks," Pattern Recognition, vol. 117, 2021.
12. H. Yang, Q. Gao, W. Xia, M. Yang, and X. Gao, "Multiview Spectral Clustering With Bipartite Graph", IEEE Trans. on Image Processing, vol. 31, pp. 3591–3605, 2022.
13. U. Von Luxburg, "A tutorial on spectral clustering," Statistics and computing, vol. 17, no. 4, pp. 395–416, 2007.
14. F. Nie, C.-L. Wang, and X. Li, "K-multiple-means: A multiple-means clustering method with specified k clusters," in Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 959–967.

15. M. E. Basbug and B. Engelhardt, "Adacluster: Adaptive clustering for heterogeneous data," arXiv preprint arXiv:1510.05491, 2015.
16. Singh, V.K., Singh, C. and Raza, H., 2022. Event Classification and Intensity Discrimination for Forest Fire Inference With IoT. IEEE Sensors Journal, 22(9), pp.8869-8880.
17. Singh, V.K., Singh, V.K. and Kumar, M., 2019. Network health monitoring of WSNs using node loss rate calculations. Wireless Personal Communications, 108(1), pp.253-268.
18. Shivhare, A., Singh, V.K. and Kumar, M., 2020. Anticomplementary triangles for efficient coverage in sensor network-based IoT. IEEE Systems Journal, 14(4), pp.4854-4863.
19. Singh, V.K., Nathani, B. and Kumar, M., 2019. WEED-MC: Wavelet transform for energy efficient data gathering and matrix completion. IEEE Transactions on Parallel and Distributed Systems, 31(5), pp.1066-1073.

Vishal K. Singh received the bachelor's degree in Information Technology, in 2010, the master's degree in Computer Technology and Application from National Institute of Technical Teachers' Training and Teachers' Training and Research, Bhopal, India in 2013, and PhD degree in Information Technology from Indian Institute of Information Technology, Allahabad, India in 2018. He is currently working as an Assistant Professor in the Department of Computer Science, Indian Institute of Information Technology, Lucknow, India and is the Head of Wireless Communications and Analytics Research Lab at IIIT Lucknow. His research interests include compressed sensing, in-network inference, markov random field, wireless sensor networks, machine learning, Internet of things and data analytics.

Gaurav Tripathi received the bachelor's degree in Information Technology in 2013, the master's degree in Computer Science and Engineering from SLIET, Longowal, India in 2018. He is now pursuing PhD from Indian Institute of Information Technology, Lucknow, India and is associated with Wireless Communications and Analytics Research Lab at IIIT Lucknow. His research interests include Machine Learning, Internet of Things, and Data Analytics.

Aman Ojha received the bachelor's degree in Information Technology in 2016, the master's degree in Computer Science from Indian Institute of Information Technology, Lucknow, India in 2021. He worked at the Wireless Communications and Analytics Research Lab at Indian Institute of Information Technology, Lucknow and currently is working in Intel, Bengaluru, India. His research interests include Machine Learning, Internet of Things, and Data Analytics.

Rajat Bhardwaj was born in Meerut, Uttar Pradesh, India. He has completed Bachelors in Technology in Computer Science and Engineering and his Master of Science in Data Science from Lancaster University. Currently, he is working as a Project Manager in the H&M Artificial Intelligence department. His area of research and interest include consumer behavior analysis, financial risk modeling, anomaly detection and recommendation systems.

Haider Raza received the bachelor's degree in Computer Science & Engineering from the Integral University, India, in 2008, the master's degree in Computer Engineering from the Manav Rachna International University, India, in 2011, and the PhD degree in computer science from the University of Ulster, Derry Londonderry, U.K., in 2016. He worked (July 2016 to Nov 2017) as a Research Officer (Data Science) in the Farr Institute of Health Informatics Research, Swansea University Medical School, U.K. He is currently a Lecturer in the School of Computer Science and Electronics Engineering, UK.

Manuscript length should be limited up to 10 pages including figures, tables, references, author's bio and photo etc. After final checks, if the typeset manuscript exceeds 06 pages length, the author will be charged (after 06) per page @ Rs.1,000/- (for Indian Authors) and 50 USD (for Foreign Authors).