Multimedia Tools and Applications (2024) 83:85523–85574
https://doi.org/10.1007/s11042-024-19917-y

An Extended-Isomap for high-dimensional data accuracy and efficiency: a comprehensive survey

Mahwish Yousaf[1] · **Muhammad Saadat Shakoor Khan**[2] · **Shamsher Ullah**[3]

Received: 7 May 2022 / Revised: 8 October 2023 / Accepted: 17 July 2024 /
Published online: 14 August 2024

Abstract
Manifold learning is a widely adopted nonlinear dimensionality reduction technique employed to discover low-dimensional representations from high-dimensional data and to explore the intrinsic data structure. It encompasses a range of nonlinear techniques, including Isomap, Local Linear Embedding (LLE), Local Tangent Space Alignment (LTSA), and Multidimensional Scaling (MDS), among others. Isomap, as a popular method in manifold learning, has exhibited promising results. However, it encounters several challenges, such as the Topological Stability Problem, Shortest Path Distance, and noise sensitivity, which compromise its accuracy and elevate its computational demands. This survey paper presents three novel existing approaches to address these challenges and enhance the performance of Isomap. Firstly, the FastIsomap method, an existing approach that combines state-of-the-art algorithms like the KD tree and NN-descent to increase graph accuracy while reducing computational cost. Secondly, the FastIsomapVis method, also an existing technique, employs a hierarchical divide-and-conquer strategy employing the KD tree and Dijkstra Buckets Double, which enhances accuracy and reduces computational time in Isomap. Lastly, the NR-Isomap method, another existing approach, used the Local Tangent Space Alignment algorithm to remove noise sensitivity and short-circuit edge problems. The existing methods improve the accuracy of Isomap but also streamline its computational requirements. Furthermore, this survey paper discusses the open issues and future research directions for Isomap and manifold learning methods. It can be a valuable resource for researchers seeking insights across diverse domains.

✉ Mahwish Yousaf
mahwish@ustc.edu.cn

Muhammad Saadat Shakoor Khan
mkhan@theory.issp.ac.cn

Shamsher Ullah
shamsher@szu.edu.cn

1 Institute of Intelligence Machine, Hefei Institutes of Physical Science, Chinese Academy of Sciences, Hefei 230027, Anhui, China

2 Key Laboratory of Materials Physics, Institute of Solid-State Physics, Chinese Academy of Sciences, Hefei 230027, Anhui, China

3 National Engineering Laboratory for Big Data System Computing Technology, Shenzhen University, Shenzhen 518060, Guangdong, China

Keywords Manifold learning · Nonlinear dimension reduction · Isomap · FastIsomap · FastIsomapVis · Randomized KD tree · NN-Descent · Local tangent space alignment · NR-Isomap

1 Introduction

Dimensionality reduction is vital in reducing redundancy and identifying fundamental sample results. This technique finds applications in data visualization, machine learning, and feature extraction. The primary objective of this technique is to minimize the amount of information with minimal data loss before proceeding with the analysis. The main challenge in dimensionality reduction is maintaining the data structure while transforming it to ensure that the essential characteristics of the original data are preserved. The technique can be classified into linear or nonlinear mapping methods. Linear mapping methods transform the data into a linear combination subspace, exchanging the original data variables for a small set of actual data variables. This allows high-dimensional data to be converted into a linear alignment of real variables for low-dimensional data representation. In contrast, nonlinear mapping methods are applied locally to nonlinear datasets, where high-dimensional and low-dimensional representations of data points are obtained by calculating the distance between actual data points [1]. Linear mapping methods cannot capture high-dimensional information on or near low-dimensional nonlinear manifolds. Nonlinear methods such as LTSA and LE are useful for small laboratory data but have not been proven effective for achieving both global and local frameworks for high-dimensional data [2, 3]. However, geodesic distance calculations are sensitive to noisy data [4], so dimensional reduction performance can be significantly reduced when using more complex and noisy input data that can damage neighboring local textures or make disjointed edges. As a result, this method cannot be utilized for linear functions or complete dataset information that contains labeled class data [5].

1.1 The definition of manifold learning

Manifold learning is a technique based on the geometry properties of data, derived from the concept of "manifold" - a locally Euclidean space shaped like a folded sheet with curves. This method encompasses clustering, dimension reduction, and semi-linear techniques and has become an efficient solution for addressing the problem of dimensionality reduction in the past decade [6–10]. The concept of manifold has been one of the most fundamental concepts in mathematics, successfully revealing the spatial form of natural phenomena. Although the traditional European research field cannot explain the nonlinear high-dimensional method, we can broaden the scope of research. The basic definitions of manifolds are provided below:

Definition 1 Topology Space: A topological space is a set of pair (X, T), where the set X is a non-empty set, and the topology T is also a subset, satisfying the following properties:

1. The union operation of any infinite element belonging to it is closed.
2. The intersection operations of the finite elements belonging to it are closed.
3. T contains the empty set 0 and X itself as an element of T.

Definition 2 Hausdorff space: Assume that X is a topological space. Let x and y be the points of X. It is said that x and y can be "separated by neighborhood." If a neighborhood U of x and a neighborhood V of y such that U and V intersect, and such a neighborhood can

separate any two points in X, we call the (X, T) Hausdorff space. Hausdorff space is also called $T2$ space or separation space.

Definition 3 Let M be a Hausdorff space if, for any point $x \in M$ in the Hausdorff space, there is an open set of a neighborhood U of x in M that is homomorphic to the dimensional Euclidean space R^m, then M is a topological manifold. $\varphi_U(U)$ is the open set in R^m, then (U, φ_U) is a coordinate of the topological manifold M. Because φ_U is a homomorphism, for any point $p \in U$, $\varphi_U : U \to \varphi_U(U)$. The coordinate data of $\varphi_U(p) \in R^m$ can be defined as the coordinate data of p in (1).

$$u^i = (\varphi_U(p))^i, \ p \in U \tag{1}$$

Where $i = 1, 2, 3, \ldots, m$ and $u^i (1 \le i \le m)$ is the local coordinate of point $p \in U$, that is, the point coordinates can describe the point in the topological manifold M in the homomorphic Euclidean space. But if you introduce different local coordinates in the neighborhood of point P in (2).

$$\varphi = U \to R^2, \psi : V \to R^2 \tag{2}$$

The function f: S → R has two different local representations in the neighborhood of point P in (3):

$$f^{\circ}\phi^{-1} : \phi(U) \to R, \ f^{\circ}\phi^{-1} : \phi(V) \to R \tag{3}$$

There may be a situation: f the two local coordinate representations have different differentiability properties. At this time, it isn't easy to define the differentiability of f at the point $p \in (U \cap V)$ based on a particular local representation. To avoid ambiguity in the definition of $U \cap V$ by using (4) and (5).

$$\psi\varphi^{-1} : \varphi(U \cap V) \to \psi(U \cap V) \tag{4}$$

With

$$\varphi\psi^{-1} : \psi(U \cap V) \to \varphi(U \cap V) \tag{5}$$

All are continuously differentiable. In this way, the differentiability of $rf^{\circ}\varphi^{(} - 1)$ can be deduced from the differentiability of $rf^{\circ}\varphi^{-1} = rf^{\circ}\varphi^{-1})(\varphi^{\circ}\varphi^{-1})$ and vice versa. It means that the two local coordinates are compatible with the differentiability of the function:$S \to R$.

If a set of local coordinates can be found whose domain covers the entire spherical surface S, they are compatible with each other in the overlapping parts of their domains. The function $f : S \to R$ can be described unambiguously as the whole spherical surface S. The above functions give a differential structure of the spherical surface S. This is the basic idea of the differential manifold.

For the intrinsic structure of the data set, it has the following characteristics:

1. Taylor's Theorem, it can be known that any differentiable function is linear in a sufficiently small neighborhood. For example, a curved manifold is spilled by small pieces of local linearity; the manifold is divided into subdivisions. The Eigen dimension of the manifold continuously changes along the topological manifold.
2. The manifold's intrinsic structure continuously changes according to the topological manifold, and the critical characteristics can only be grasped locally.

Definition 4 Let $Y \in R^d$ be a low-dimensional manifold, and the smooth embedding function is $f : Y \to R^D$, where $D < d$. The data set y_i is generated data set randomly, and after f is mapped into the data $x_i = f(y_i)$ in the observation space. Manifold learning is

to reconstruct f and y_i under the condition of a given observation sample set x_i. According to the above definition, this is the basic principle of the manifold learning method of the non-linear dimensionality reduction method. Manifold learning methods handle dimensionality reduction and discover the intrinsic geometry structure from high-dimensional to low-dimensional space. The mathematical problem is as follows: a given $x_1, \ldots, x_n$ of n data points in $R^d (d < D)$, search a set of points $y_1, \ldots, y_n$ in R^d, such that $y_1$represent the x_1. In manifold learning methods, where $x_1, \ldots, x_n \in M$ and M is an embedded manifold in $R^d (d < D)$ [11].

1.2 Comparison with previous reviews

High-dimensional data can be difficult to manage due to its large size and varying dimensions. In many fields, such as machine learning, data mining, computer vision [2, 12, 13], and information visualization [14–16], it is crucial to find low-dimensional representations of high-dimensional data while minimizing data loss [1]. Dimensionality reduction techniques can be categorized as linear or nonlinear methods. Linear mapping methods include Principal Component Analysis (PCA) [17], Multidimensional Scaling (MDS) [13], Independent Component Analysis (ICA), Singular Value Decomposition (SVD), CUR matrix decomposition, Compact Matrix Decomposition (CMD), and Non-negative Matrix Factorization (NMF). Nonlinear methods include Kernel PCA, Laplacian Eigenmaps (LE) [12], Isomap [6], FastMap, Locally Linear Embedding (LLE) [7], diffusion maps [18–20], Hessian Eigenmaps (HE) [21], Autoencoders [22], and t-SNE [3]. However, these nonlinear methods are susceptible to the out-of-sample problem [23, 24].

H. Choi and S. Choi proposed a kernel Isomap method that effectively addresses outliers, noise, and topological instability issues in the Topological Stability Problem (TSP) [25]. They employed network flow [26, 27] to reduce the impact of outliers based on the topological structure, which was further improved by Choi et al. with a kernel method [28, 29]. Mohammad et al. presented a new algorithm that effectively handled the short-circuit bridge problem in topological instability by finding the K nearest neighbor [30]. Mahwish et al. proposed the FastIsomap method, which tackled both the high computational cost and topological stability issues of Isomap [31].

To address the problem of noise sensitivity in dimensionality reduction, Hong Chang et al. [32] introduced the robust LLE method, which utilizes a weighted robust PCA algorithm to eliminate noise and outliers in the dataset, followed by applying the weighted LLE method to the cleaned dataset. However, this method has a limitation in that it does not remove noise and outliers in the dataset but only improves the robustness of LLE. To overcome this issue, Bo Li et al. [27] proposed an extended Isomap method that employs robust PCA and box statistics [33] for noise and outlier removal. The topological structure can be better preserved by denoising the dataset with the proposed algorithm. Bo Jin et al. [34] presented an improved neighborhood selection algorithm for the Isomap and LLE methods. The algorithm selects the best neighborhood zone based on the relationship between the neighborhood datasets. To address the high time complexity, sensitivity to outliers, and short-circuit edges problem in topological instability, Chen and Mao [35] proposed the LP center-based Isomap (LP-Isomap) method. They utilized the k-nearest Center (k-NC) and Linear Patch (LP) methods to solve these problems. Furthermore, Chun Du et al. proposed [36] a Robust Isomap based on the Neighbor Ranking Metric (NRM) method, which can effectively remove the outliers from the selected neighborhood data points and construct the graph. Kouropteva et al. [37, 38], Shao et al. [39], and Saxena et al. [40] proposed two methods for LLE and Isomap, called

the selection of optimal parameter values method and an integrated approach to address the short-circuit edges and noisy dataset problems. Guy et al. [41] presented Topologically Constrained Isomap Embedding (TCIE) for removing and detecting in-consistent distance problems of Isomap. Yves et al. [42] suggested the Density-based Graph Denoising (DGD) method for detecting shortcuts and noisy datasets. It can construct a shortcut-free graph for noisy datasets. All the previous techniques are performed well for outliers, noises, and short-circuit edge problems but are a little efficient. Although these techniques focused on the noise and short-circuited edges problems, neither accuracy nor time speed.

Various approaches have been proposed to tackle the shortest path problem in manifold learning. W. Hong et al. [43] suggested two approaches, namely the multi-class multi-manifold-Isomap (MCMM-ISOMAP) and Isomap for classification (ISOMAP-C), to overcome the slow computation of the Isomap algorithm. Taiguo et al. [44, 45] introduced an improved Isomap method based on the Fib-Dij algorithm, which can effectively remove nearest neighboring graphs and high-dimensional data points that may negatively affect the Isomap graph [4, 25]. Ying-K Lei et al. [46] used the greedy approximated algorithm of Minimum Set Coverage (MSC) [47] to handle both the shortest path and MDS problems. Mahwish et al. [48] proposed the FastIsomapVis method, which uses a hierarchal divide, conquer, and combine approach to solve the shortest path and the lack of topological stability problems. Additionally, Li and Chao [49] suggested a straightforward method with residual variance to address the computation time and neighborhood size problems in the shortest path problem. Finally, Zhao Zhang et al. [50] proposed the Marginal Isomap (M-Isomap) method, which calculates the shortest path distance between pairwise data points based on two constrained types: Cannot-Link and Must-Link. Amir Najafi et al. [51] proposed the path-based Isomap, which uses pairwise geodesic distance instead of a path mapping algorithm for low-dimensional embedding optimization.

1.3 Contributions

Our survey paper aims to address the three major challenges of Isomap: high accuracy, high performance, and computational time cost, by examining the progress made in the last three years in Extended-Isomap techniques. We extensively review three Isomap problems and propose existing techniques to tackle them. In this survey paper, we examine Isomap within the context of both linear and nonlinear manifold learning techniques. We assess the efficacy of the established FastIsomap method in addressing the Topological Stability Problem (TSP) by utilizing the KD tree and NN-Descent algorithms. Additionally, we evaluate the performance of the existing FastIsomapVis method in mitigating TSP and shortest path challenges, employing a hierarchical divide, conquer, and combined strategy. Furthermore, we gauge the established NR-Isomap method's effectiveness in handling noise-related issues. Finally, we present a comparative tabulation showcasing the performance of the recommended current approaches, leveraging various algorithms and strategies. Our contributions are summarized as follows:

1. We reviewed the Isomap method in linear and nonlinear Manifold Learning techniques.
2. Conducted an extensive review of the proposed existing FastIsomap method for Topological Stability Problem (TSP) with KD tree and NN-Descent algorithms, which improved the accuracy and computational time cost of Isomap.
3. We reviewed the proposed existing FastIsomapVis method for TSP and shortest path problems with a hierarchical divide, conquer, and combined approach, which improved the accuracy and computational time cost of Isomap.

4. We comprehensively reviewed the proposed existing NR-Isomap method for noise sensitivity and short-circuit edge problems with the local tangent space algorithm, which improved the noise reduction.
5. We evaluated the three existing methods tabularly according to different algorithms and techniques.

1.4 Paper organization

The rest of this survey paper is organized as follows in Fig. 1. We present several representative manifold learning algorithms, including linear and nonlinear methods, in Section 2. Section 3 discusses the details of the Isomap algorithm, its problems, and its main challenges. Section 4 discusses the details of the existing FastIsomap method and experimental results. Section 5 comprehensively discusses the existing FastIsomapVis, including the hierarchical divide and conquer combined approach and detailed experimental results. Section 6 discusses the existing NR-Isomap with the LTSA, Dijkstra, and MDS algorithms.

Moreover, we describe the results and discussion on real-world datasets. Section 7 discusses the open problems and future research direction for the Isomap algorithm and manifold learning method. Finally, in Section 8, the paper is concluded.

2 Several representative manifold learning algorithms

In recent years, many research results have been produced in manifold learning. A large number of research results have been produced in the area of manifold learning in recent years. The Representative Stream Shape learning methods include Isometric Feature Mapping (ISOMAP) [6], Locally Linear Embedding (LLE) [7, 52], Laplacian Eigenmaps (LE) [8], Hessian-based Locally Linear Embedding (HLLE) [21], Local Tangent Space Alignment (LTSA) [53], and so on. This section reviews linear and nonlinear techniques described below, such as PCA, LDA, Classical Isomap, MDS, LLE, and Hessian Embedding methods.

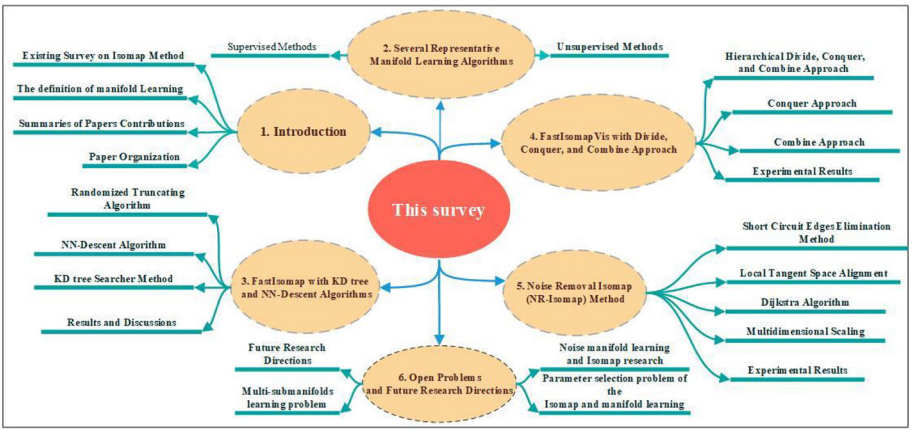

Fig. 1 The structure of the paper

2.1 Linear methods

The linear method means that the low-dimensional data is obtained after data dimensionality reduction. It can still maintain the linear relationship between high-dimensional data points. For example, an N high-dimensional data point with D is $X_1, X_2, \ldots, X_N$, and the dimension of the data obtained after data dimensionality reduction is still ($d \leq D$). Then the result after dimensionality reduction is $Y_1, Y_2, \ldots, Y_N$, If there is a linear mapping F make such that $F(Y_i) = X_i, i = 1, 2, 3, \ldots, N$. Then, this process of dimensionality reduction is called linear dimensionality reduction. We have described two linear methods are given below:

2.1.1 Principal component analysis

PCA is one of the most classic linear dimensionality reduction methods. It is also called the Hotelling Transformation. The researcher's Hotelling was proposed in 1993, and PCA was proposed in 1999, which uses the variance of statistics as the standard amount of information in a data set. It believes that the greater variance value provides more information and vice versa. PCA uses a linear combination of the original components to construct a large variance and more informative principal components to reduce the dimensionality. The basic idea of the PCA is reconstruction to reduce the dimensionality of the data. It needs to choose the direction projection that maximizes the variance after projection and protects the data's information as much as possible. Finally, it can be obtaining the problem of calculating eigenvalues and eigenvectors. So, its implementation is straightforward, without iteration, to achieve the optimal global solution. PCA calculated eigenvalues by performing the eigendecomposition on the data covariance matrix [54]. Suppose there are M data $X_1, X_2, \ldots, X_M$ which assumes $X_i \in R^M$, these data It is required to be zeroed by the center $\sum_{i=1}^{M} X_i = 0$. Define the covariance matrix in (6)

$$\frac{1}{M} \sum_{i=1}^{M} X_1 X_M^T = C \tag{6}$$

Because the covariance matrix C is a positive semi-definite matrix, we diagonalize it and get $C = V \wedge V^T$ the diagonal matrix takes eigenvalues, and V is the matrix that takes the corresponding eigenvectors [1]. Here is the diagonal (7) is:

$$\wedge = diag(\lambda_1, \lambda_2, \ldots, \lambda_n), VV^T = 1 \tag{7}$$

In Fig. 2, we have shown the visualization form of the PCA method in two dimensions. We can see that the yellow color indicates the setosa class, the orange color shows the Versicolor class, and the pink color shows the Virginica class. However, the behavior of the setosa class differs from the other two categories, and this class overlaps each other according to dimensions.

2.1.2 Linear discrenment analysis

LDA is a method for feature extraction. It is different from the idea of principal component analysis. The goal of LDA is to find the optimal projection direction. The maximum is the ratio between all sample data projections' inter-class and intra-class dispersion. Because of this, LDA is much better in classification, so LDA is often used for classification. LDA is one of the best methods of feature extraction. LDA's fundamental concept is to reduce dimensionality when preserving class information as much as possible [55]. LDA is provided

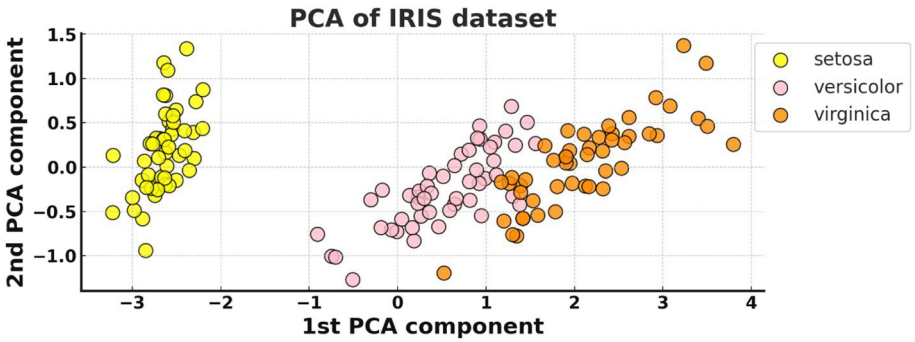

Fig. 2 Visualization form of the PCA Method with Iris dataset

with the solution of eigenvalues and eigenvectors, so its implementation is straightforward. It only needs to iterate to achieve the optimal global solution. LDA considers the problem from the perspective of data classification; it is a linear pattern classification method. The LDA has performed the five steps [56] as given bellows:

1. Calculate the d dimensional vector means by using (8); M_i is used for different classes from the dataset.
2. Calculate the scatter matrices within and between the classes using (9).

$$S_w = \sum_{i=1}^{c} S_i \tag{8}$$

$$S_b = \sum_{i=1}^{c} N_i (M - M)(M_i - M)^T \tag{9}$$

   where S_i is the scatter matrices within a class, M is the overall mean vector, and N_i is the size of every scatter matrices between classes.
3. Resolve the problem of the eigenvalue matrix $S_w^{-1} S_b$. The relationship between eigenvalues and eigenvectors represents the distortion of the linear transformation. The eigenvector shows the direction, and eigenvalues show the magnitude of the distortion.
4. Choose the new feature space's linear discernment (LD). Firstly, this process sorted the eigenvectors in descending order of eigenvalues. Also, select the eigenvectors K within the largest eigenvalues, then make the eigenvector matrix $W_{(d \times k)}$.

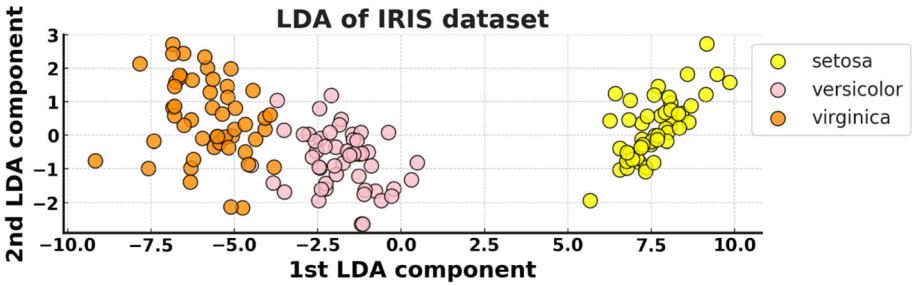

Fig. 3 Visualization form of the LDA Method with Iris dataset

5. Transformed the samples in the new subspace samples through this equation $Y = X \times W$, where X is the $n \times d$ sample matrix, and Y is the transformed subspace matrix [1].

Figure 3 shows the visualization form of the LDA method in two dimensions. We can see that the yellow color indicates the setosa class, the orange color shows the Versicolor class, and the pink color shows the virginica class. However, the behavior of the setosa class differs from the other two categories, and this class overlaps each other according to dimensions. Also, the setosa class performs well in both LDA and PCA and differs from the others. Moreover, the LDA method is better executed than the PCA method, and the Versicolor class and virginica class between overlapping are minimum rather than the PCA method.

2.2 Nonlinear methods

The emergence of manifold learning opens a new way of thinking about nonlinear data processing. The manifold theory believes that all high-dimensional data has transformed into a low-dimensional space. It reduces data dimensionality by maintaining a particular topological structure of the source data. The linear and nonlinear dimensionality reduction methods are based on manifold learning. The linear dimensionality reduction method is based on local variables. A fundamental assumption in manifold learning is that the processed data comes from a potential or a possible manifold. Different manifold learning methods have additional requirements for the properties, resulting in different hypotheses under the manifold theory's conditions. We have discussed three nonlinear methods given below:

2.2.1 Multidimensional scaling

MDS is a statistical approach that can visualize the dissimilarities and similarities. It can be applied for multidimensional data analysis in various research areas, such as psychology, economics, etc. MDS is a nonlinear dimensionality reduction method that preserves the similarity between data points. MDS can be divided into multidimensional metric scaling and non-metric multidimensional scaling. MDS is a low-dimensional representation of data using the distance between data points. It can be preserved the pairwise distances in low-dimensional space. After visualizing the data, similar data are close to each other, and dissimilar data are far away. The MDS algorithm's basic idea is to calculate the coordinates of points through a matrix describing the degree of difference between points. [57].

Suppose that N high dimensional vector $x_i \in R^N, i = 1, 2, \ldots, m$ corresponding to the low-dimensional vector $y_i \in R^d, d < N$. The Euclidean distance between vectors x_i, x_j and calculated the dissimilarities; denote as δ_{ij}. The lower-dimensional distance is often the Minkowski distance between y_i and $y_i - d(y_i, y_j), i, j = 1, 2 \ldots, m$ by using (10) [58].

$$d_p(y_i, y_j) = \left(\sum_{k=1}^{m} |y_{ik} - y_{jk}|^p \right) \tag{10}$$

We have calculated the Euclidean distance while $p = 2$.

$$d_p(y_i, y_j) = \left(\sum_{k=1}^{m} |y_{ik} - y_{jk}|^p \right) \tag{11}$$

We have calculated the Euclidean distance while $p = 2$.

Figure 4 shows the visualization form of the MDS method in two dimensions. We can see that the yellow color indicates the setosa class, the orange color shows the Versicolor

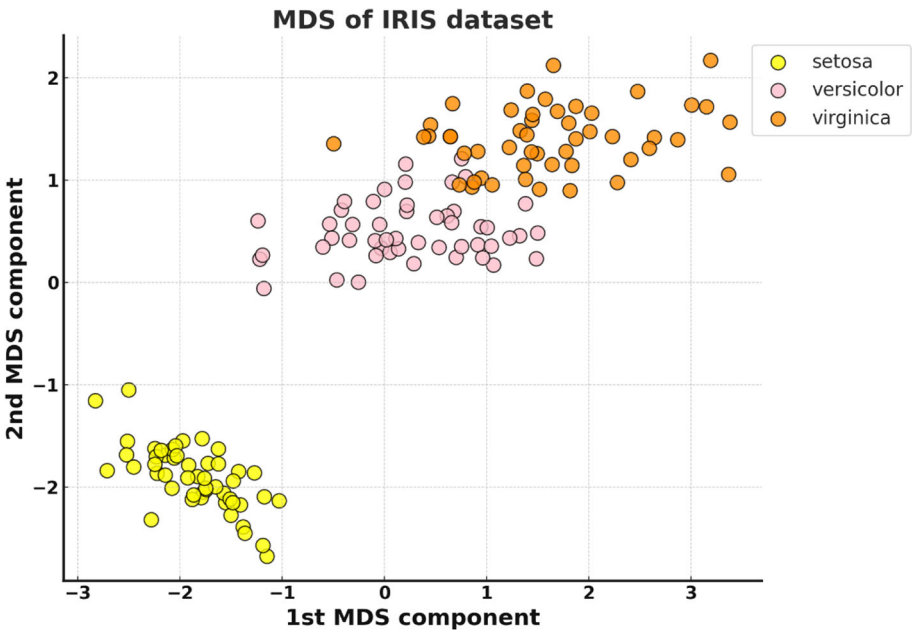

Fig. 4 Visualization form of the MDS Method with Iris dataset

class, and the pink color shows the virginica class. But the performance of the setosa class is changed from the other two categories. This class overlaps with each other according to dimensions.

2.2.2 Locally linear embedding

LLE's basic idea is to construct a local linear plane between the sample point and its neighboring points and establish an optimized objective function. In other words, a sample point can be linearly reconstructed from its neighboring points. Moreover, the reconstruction weight minimizes the linear reconstruction error between the sample and neighborhood points. Repeat the above optimal reconstruction weight for each sample point, keeping the reconstruction relationship between each sample point and its neighbor points in the low-dimensional space unchanged. Assume that the embedding in the low-dimensional space is locally linear. The high-dimensional vector's embedding in the low-dimensional space can be obtained by minimizing the reconstruction error. The minimum reconstruction error weight in LLE can maintain the geometric properties of the local neighborhood of the data; that is, the minimum reconstruction error remains unchanged for the rotation and scaling of the data [52]. The LLE algorithm first finds its neighborhood at each sample point and then calculates the minimum reconstruction weight by solving the constrained least-squares optimization problem by (12)

$$\epsilon_i = min \| \sum_{j=1}^{k} w_{ij} x_j \| \tag{12}$$

where K is the neighborhood data points. When solving this least-squares problem, LLE transforms it into a linear equation system that may be singular and introduces a small

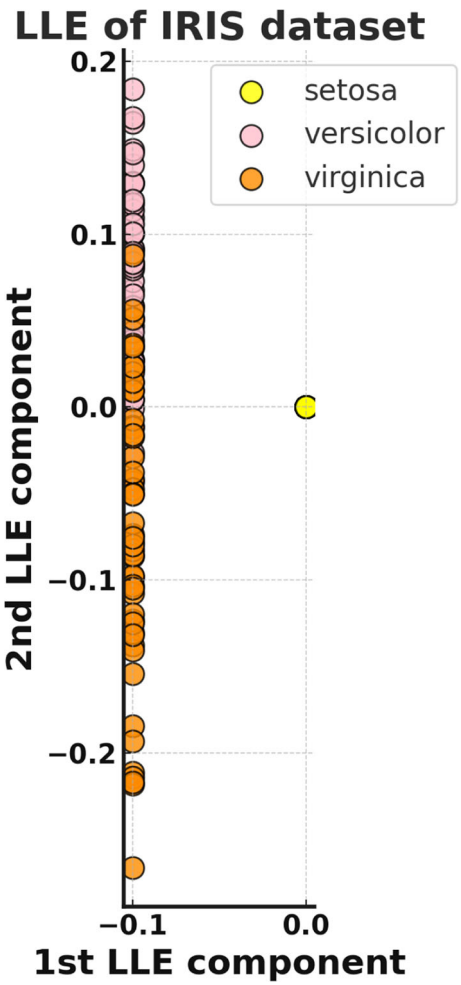

Fig. 5 Visualization form of the LLE Method with Iris dataset

regular factor to ensure the non-singularity of the coefficient matrix of the linear equation system. After obtaining the reconstruction weights, using these reconstruction weights can construct a sparse matrix from (13)

$$M = (1 - w)^T (1 - w) \tag{13}$$

LLE obtains the data's low-dimensional embedding by solving the eigenvectors corresponding to this sparse matrix's smallest eigenvalues. Since the eigenvector corresponding to the eigenvalue is a vector, it cannot be used for dimensional embedding.

Figure 5 shows the LLE method's visualization form in four dimensions and K nearest neighbor=15. We can see that the yellow color indicates the setosa class, the orange color shows the Versicolor class, and the pink color shows the virginica class. However, the performance of all categories is changed according to dimensions and neighborhood size and represents the noisy behavior. The LLE method is a clear cluster in Fig. 6. Therefore, all classes are overlapped with each other.

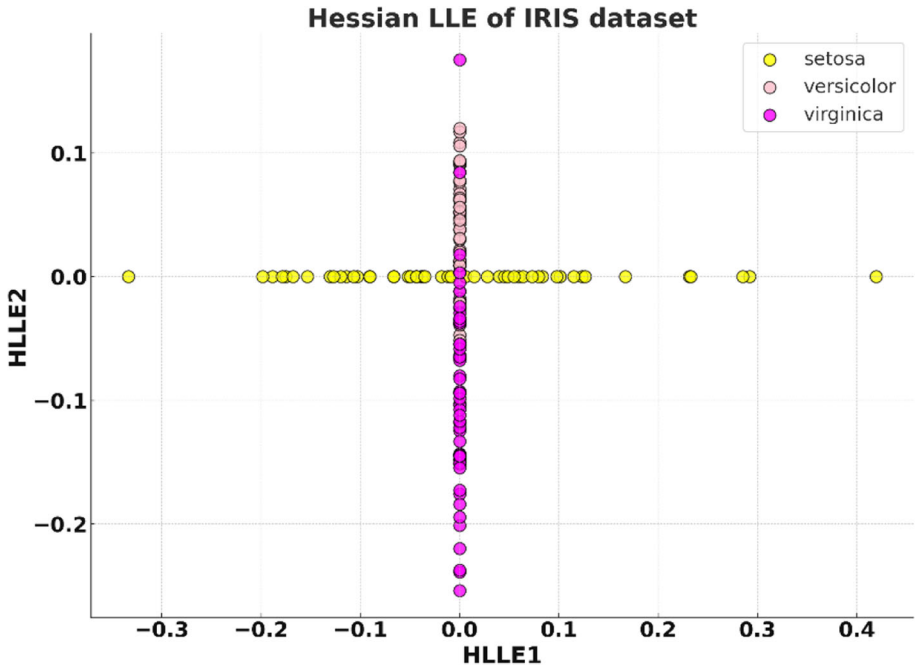

Fig. 6 Visualization form of the HE Method with Iris dataset

2.2.3 Hessian embedding

The Isomap algorithm is applied to the iris dataset in the previously described manifold learning algorithm. The sample data points are uniformly clustered and overlap with each other. When these conditions are not met, the Isomap method is not sufficient and valuable. Donoho et al. [21] proposed a Hessian embedding method to deal with uneven data sampling. The Hessian embedding method is also called the Hessian LLE method. Figures 5, 6, and 7 show the mapping results of LLE, Isomap, and Hessian embedding methods to data with clusters; Hessian proved that the feature mapping method is helpful in the data processing. The Hessian embedding method and the Laplacian eigenmaps method's theoretical framework are very similar; the difference is that the Hessian operator is used instead of the Laplacian operator. Therefore, the Hessian embedding method aims to find such a mapping, $f = M \rightarrow \Re^D$. Where $f \in C^2(M)$ the hessian operator uses the (14) for local coordinate definition:

$$H_f^{tan}(m)_{i,j} = \frac{\delta^2 g(x)}{\delta x_i \delta x_j} | x = 0 \tag{14}$$

$$H(f) = \int_M \| H_f^{tan}(m) \|_F^2 d_m \tag{15}$$

from (15), where m is a point on the observation manifold, d_m is a strictly positive probability measure on manifold M, $H(f)$ is a measure of the average curvature of the function plant on manifold M, $g(x) = f(m)$ is a function $g = U \rightarrow \Re^d$, where U is $\Re^d$ a neighborhood of the origin, m is a neighbor of observation space m.

Figure 6 shows the Hessian method's visualization form in four dimensions and K nearest neighbor=15. We can see that the yellow color indicates the setosa class, the orange color shows the Versicolor class, and the pink color shows the virginica class. The performance of all categories is changed according to dimensions and neighborhood size and overlaps with each other. However, the Hessian embedding method performs better than the Isomap and LLE methods.

3 ISOMAP

Isomap is a variant of the MDS metric that uses nonlinear data for geodesic distance. The basic idea of Isomap is to preserve the geometry structure of the data and obtain a geodesic distance between all pairs of data points. The geodesic distance is separated into two steps: the neighboring data points and faraway neighboring data points. The geodesic distance is calculated as the shortest path distance for nearby and far neighboring points [27, 44, 45]. The Algorithm 1 of Isomap is given below:

Figure 7 shows the visualization form of the Isomap method in two dimensions and K nearest neighbor=10. We can see that the yellow color indicates the setosa class, the orange color shows the Versicolor class, and the pink color shows the virginica class. However, the performance of the setosa class is changed from the other two categories, and this class overlaps with each other according to dimensions and represents the noisy form. Therefore, the Versicolor and virginica classes overlap according to K nearest neighbor size.

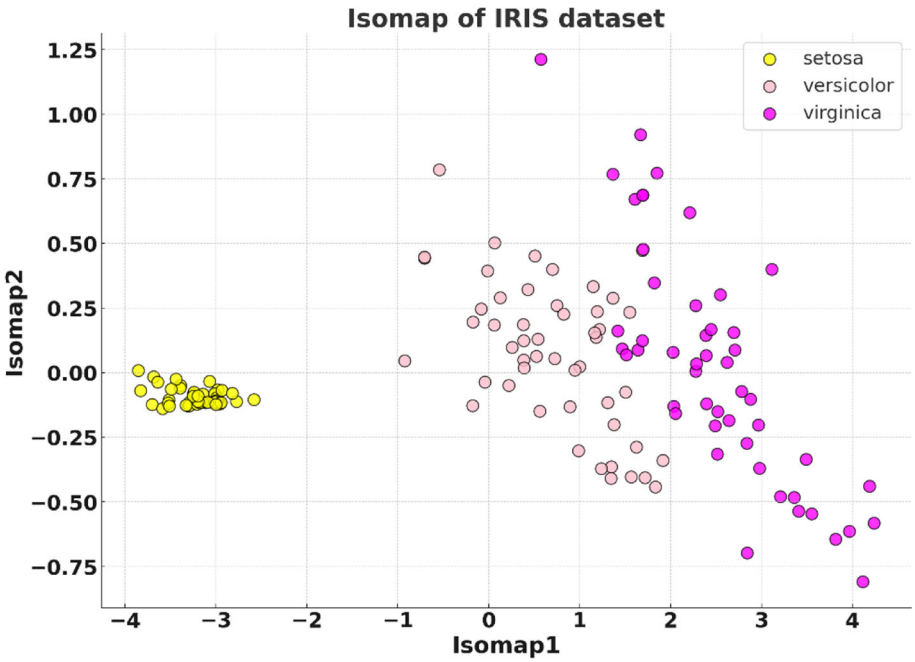

Fig. 7 Visualization form of the Isomap Method with Iris dataset

Algorithm 1 Isomap.

Require: Number of nearest neighbor data points: K, distance: d
Ensure: KNN graph: G

1: **Build neighbourhood graph** G—Firstly, build the k nearest neighbor (KNN) graph G of manifold learning based on the Euclidean distance d between two data points in the input space X_i and X_j, i.e., $d = (X_i, X_j) = \|X_i - X_j\|$ [27].
2: **Calculate the shortest distance**—When building the neighborhood graph G, then calculate the geodesic distance matrix between sub-neighborhood faraway data points and compute the shortest path distance between any two data points X_i and X_j is the graph G by Dijkstra and Floyd's algorithms [59].
3: **Build a d-dimensional embedding graph**—Isomap uses the MDS algorithm to compute the low d-dimensional embedding of the data points and make the geodesic distance dense matrix [1].

3.1 The problem

Our primary motivation for this study is to investigate the problems associated with the Isomap algorithm. One core issue in Isomap is the lack of a Topological Stability Problem (TSP) in the neighborhood graph G, which leads to incorrect links within the graph [26, 44, 45]. Another problem is the algorithm's slow performance and high computational time cost, primarily due to its dependency on the Shortest Path Distance (SPD) and the Dijkstra algorithm. As the number of nearest neighbor data points (K) increases, Isomap encounters two computational challenges: computing SPD using the Dijkstra algorithm and calculating eigenvalues using the MDS algorithm. Notably, Isomap exhibits a notable weakness in computational time complexity when calculating the geodesic distance for each neighborhood, particularly with increasing K values. The third issue in Isomap is related to topological stability, which introduces problems such as noise sensitivity and short-circuited edges. Noisy data in Isomap can lead to TSP, wherein short-circuited edges directly connect two sub-manifold data points. This phenomenon poses challenges to the algorithm, particularly in the presence of large nearest neighborhoods and dependency on the neighborhood size.

3.2 Main challenges

The Isomap algorithm faces significant challenges in achieving high accuracy, high performance, and managing computational time costs. These challenges arise due to factors such as the selection of the neighborhood size parameter (K), the presence of high-dimensional and large-scale datasets, and the limited capability of Isomap to deliver highly accurate results for datasets containing millions of data points across hundreds of dimensions. Additionally, Isomap struggles to effectively mitigate noise in such datasets as the choice of the neighborhood size (K) affects its performance. Consequently, processing high-dimensional and large-scale datasets with Isomap becomes slow and time-consuming. While existing research and scholarly efforts have primarily focused on addressing the high performance and time costs associated with Isomap, less attention has been given to achieving high accuracy. Thus, this survey paper primarily emphasizes addressing the challenges of achieving high accuracy, high performance, and managing computational time costs within the Isomap algorithm.

4 FastIsomap with KD tree and NN-Descent algorithms

In this section, we review the existing FastIsomap [31] method for the topological stability problem of the Isomap method. We discuss the details of the KD tree and NN-Descent algorithms and represent the complete results of the existing FastIsomap method.

Given high-dimensional and large datasets $X = x_i \in R^d (i = 1, 2, \ldots, N)$, we signify each dataset x_i with a low-dimensional vector $Y_i \in R^d$ So the value of d is 2 or 3. The Isomap algorithm is ineffective for reducing high-dimensional and large datasets to lower dimensions. We employ the Euclidean distance metric, denoted as E, for similarity calculations among data points through a square matrix to address this. This approach yields superior results compared to geodesic distances. Building upon this, the existing FastIsomap method leverages Randomized Truncating Trees (KD tree) and NN-Descent algorithms to tackle issues like incorrect links (topological stability) efficiently. FastIsomap operates in three distinct phases.

1. **Phase-1:** We divided the data points into small ones and found the K nearest neighbor using a KD tree algorithm. Also, divide the graph into a subgraph.
2. **Phase-2:** We have combined the subgraph data points and purified graph data using the NN-descent algorithm.
3. **Phase-3:** We have used the KD tree searcher algorithm to calculate accuracy [31].

We have shown the general framework of our existing FastIsomap [31] method in Fig. 8.

4.1 Randomized truncating (KD tree) algorithm

Phase 1 of FastIsomap utilizes the KD tree algorithm, which plays a crucial role in improving the performance of the Isomap method. The KD tree algorithm facilitates better initialization of data points by dividing them into two sub-data points based on the variance at each binary tree level [60]. To enhance the efficiency of the K nearest neighbor (KNN) search in high-dimensional space, Silpa-Anan et al. [61] introduced the randomized KD trees algorithm. This approach recursively splits the data points according to predefined dimensions and randomly selects from small data sizes with the highest variance. Additionally, the KD tree includes a KD tree searcher method for determining the K nearest neighbor values.

We have used the KD tree algorithm to construct a tree in Algorithm 2. In lines (1-7), the KD tree creates the tree node and point set and randomly picks the d data dimensions. We have computed the median of the data points on the d data dimension. According to the median, we have evenly split the tree nodes into two halves nodes, i.e., left-half and right-half. In lines (8-9), we have called the construct-tree function two times according to left-half and right-half nodes. In lines (12-14), Algorithm 2 calls the construct-tree function recursively and adds node according to the datasets D and Root-Node. KD tree Algorithm 2 is given below:

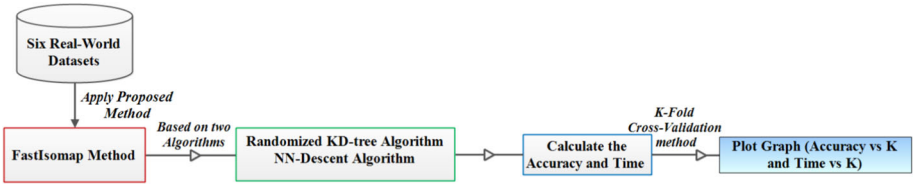

Fig. 8 General Framework of the Existing FastIsomap method

Algorithm 2 KD tree algorithm of FastIsomap.

Require: Number of trees: T, the number of node points: K, Dataset: D, and Approximate KNN graph: K
Ensure: A Randomized KD tree set: R and an Approximate KNN graph: G
1: Function Construct-Tree (NODE, POINT-SET)
2: **if** (size of POINT-SET $\leq$ K) **then**
3: Return
4: Select Randomly data d dimension d
5: Compute median over the POINT-SET on d Dimensions of data.
6: Split the POINT-SET equally into two sections:
7: LEFT-HALF'S and RIGHT-HALF'S, according to the median.
8: Construct-Tree (NODE.LEFT-CHILD,LEFT-HALF'S)
9: Construct-Tree (NODE.RIGHT-CHILD, RIGHT-HALF'S)
10: **end if**
11: Return
12: **for** (i $\leftarrow$ 1, …, T) **do**
13: Construct-Tree $(ROOT - NODE_i, D)$
14: Add $(ROOT - NODE_i)$ to R
15: **end for**
16: end Function Construct-Tree

4.2 NN-descent algorithm

In their work, Wei Dong et al. [62] introduced the NN-descent algorithm, which focuses on refining the resulting graph. The key concept behind the NN-descent algorithm is that a neighbor of a neighbor is likely to be a neighbor as well. The existing FastIsomap [31] used the KD tree algorithm to construct an approximate K nearest neighbor (KNN) graph in phase 1. However, since the accuracy of the datasets is not exceptionally high, we employ the NN-descent algorithm specifically for datasets with higher accuracy. The execution of the NN-descent algorithm commences with the KNN graph. It iteratively enhances the dataset's accuracy by identifying neighboring nodes associated with the current tree node. This iterative process continues based on the specified K values until the graph is refined, achieving high accuracy. As a result, it can obtain 100% accurate results within a limited number of iterations for the KNN graph.

We have used the NN-descent algorithm to refine the KNN graph in phase 2. Algorithm 3, we call KD tree Algorithm for the refinement of Algorithm 2 via NN-descent algorithm. In lines (3-4), the graph G initializes and randomly makes a test list of KNN for a reverse neighbor of data point U. In lines (6-7), we have checked the different reverse pairs of neighbor $U's(v, w)$ and RNN list and then estimated the distances of neighbors (v, w). In lines (8-12), update the KNN distance of neighbors (v, w) list according to w and v and repeat these iterations according to the K values. NN-descent Algorithm 3 is given below:

4.3 KD tree searcher method

This section used the KD tree searcher algorithm to calculate the existing FastIsomap method's overall accuracy. It gathers the results from the NN-descent Algorithm 3 and calculates the accuracy via the search model object method. The KD tree algorithm repetitively splits N by K data points and the N data points in K dimensional space into a binary tree mode. KD searcher Algorithm 4 is given below:

When we make a KD tree searcher model object, it identifies the stored tree. The KD tree searcher method has two methods, Knnsearch and Rangesearch, for finding KNN values. In phase 3, we use the Knnsearch method in our existing proposed method [63, 64].

Algorithm 3 NN-descent algorithm of FastIsomap.

Require: $X = i = 1, 2, \ldots, N$, Number of trees: T, Dataset: D, Reverse neighbor of a data point: U, Distance function: dist, and Neighbourhood size: K
Ensure: An Approximate KNN graph: G Using Algorithm 1 to Construct-Tree R shows the tree input.
1: Neighbor exploring (NN-Descent)
2: **for** (data-point U ∈ D) **do**
3: Initialization of graph G by randomly creating the test list of KNN for U with a given distance of ∞.
4: **end for**
5: **for** (data-point U ∈ D) **do**
6: Check different pairs of U's neighbors (v,w) in U's KNN and RNN list and calculate the distance dist(v,w).
7: Use (v, dist(v, w)) to update w's KNN list
8: Use (w, dist(v, w)) to update v's KNN list
9: **end for**
10: Until G meets
11: Return G

Algorithm 4 KD tree searcher algorithm of FastIsomap.

Require: NSMethod: Nearest Neighbour Search Method, A: Accuracy, and CP: CorrectRate
Ensure: Accuracy and Time
1: Start Time
2: D = Createns(G, 'NSMethod', 'kDtree', 'Distance', 'euclidean')
3: KDTreeSearcher properties (K)
4: **for** (K ← 1, …, 5) **do**
5: A = Knnclassify (D, D, D, K,'eucli')
6: CP = Classperf (D,A)
7: Time (K) = End time
8: K (K) = CP.CorrectRate
9: Return CP
10: **end for**

4.4 Results and discussions

This section evaluated our existing FastIsomap for publicly accessible six large and high-dimensional datasets such as Facebook, Twitter, Live-Journal, YouTube, Orkut [65], and SIFT1M [66]. Our experimental results evaluate the efficiency of the existing FastIsomap with KD trees and NN-descent algorithms. The details of the six datasets are listed in Table 1.

4.4.1 KNN classier method

In our experimental evaluation, we employed various classification techniques to measure the accuracy of the existing FastIsomap and Isomap methods. The commonly used classification

Table 1 Characteristic of Datasets

Datasets	Data	Dimensions	Categories
Facebook	4,039	40	10
Twitter	81,306	40	973
SIFT1M	100,000	128	1000
YouTube	1,134,890	150	5000
Orkut	3,072,441	150	5000
Live-Journal	3,997,963	100	5000

Table 2 Comparison Accuracy of KD Tree Method with Facebook, Twitter, SIFT1M, Live-Journal, YouTube, and Orkut Datasets

K	Facebook	Twitter	SIFT1M	Live-Journal	YouTube	Orkut
1	1.00	1.00	1.00	1.00	1.00	1.00
2	1.00	1.00	1.00	1.00	1.00	1.00
3	0.99	0.99	0.97	0.98	0.92	0.96
4	0.99	0.99	0.94	0.95	0.87	0.89
5	0.99	0.99	0.88	0.94	0.80	0.86
6	0.99	0.99	0.84	0.91	0.76	0.79
7	0.98	0.99	0.80	0.89	0.71	0.76
8	0.98	0.98	0.77	0.87	0.67	0.71
9	0.98	0.98	0.70	0.84	0.63	0.69
10	0.98	0.98	0.70	0.84	0.61	0.66
11	0.97	0.97	0.66	0.83	0.59	0.64
12	0.97	0.97	0.65	0.81	0.58	0.61
13	0.96	0.96	0.60	0.80	0.56	0.58
14	0.96	0.96	0.53	0.78	0.55	0.56
15	0.95	0.95	0.53	0.76	0.53	0.55

techniques include the K nearest neighbor (KNN) classifier [67, 68], Support Vector Machines (SVM) [69, 70], Backpropagation Neural Network (BPNN) [71], and Decision Tree (DT) [72]. The existing FastIsomap [31] assessed the accuracy through the KNN classifier with the K-Fold Cross-Validation (KFCV) technique [73]. KFCV enables the evaluation of multiple parameter values to identify the optimal choice. We also employed the KFCV method to determine the accuracy of the KD tree construction and the refinement algorithms, such as NN-Descent, used to create the KNN graph. In KFCV, the available data points are randomly divided into N partitions of equal size, and the process is repeated N times. The accuracy ($Accuracy$) is calculated based on the number of correct classifications (C) over the N data points. The accuracy is computed using the formula defined in (X).(16):

$$Accuracy = \frac{P}{C} \tag{16}$$

4.4.2 Experiments on KD tree method

Tables 2 and 3 have shown the calculated accuracy and time of Facebook, Twitter, Live-Journal, YouTube, Orkut, and SIFT1M datasets from (16).

Figures 9 and 10 presented the performance of the KD tree algorithm of the KNN graph w.r.t accuracy and time cost. In Tables 2 and 3, values to calculate six datasets' time and accuracy. The Facebook dataset has 4,039 data points and provides 95% accurate $K's$ value (1 to 15). The Twitter dataset contained 81,306 data points; it is complicated to construct the KNN graph at 95% accurate results via the KD trees algorithm. Facebook and Twitter datasets have the same accuracy for the KD tree algorithm, but the time is different because Twitter datasets are vast and higher than the Facebook dataset. The SIFT1M dataset contained 100,000 data points and presented 53% accurate results within less time. YouTube dataset has 1,134,890 data points and displays 53% accurate results on the value of $K = (1 \quad to \quad 15)$ but gets the high time rather than the SIFT1M dataset. The Orkut dataset contained 3,072,441

Table 3 Comparison time of KD tree method with Facebook, Twitter, SIFT1M, Live-Journal, YouTube, and Orkut Datasets

K	Facebook	Twitter	SIFT1M	Live-Journal	YouTube	Orkut
1	271.76	366.11	5.09	11.96	110.55	17.28
2	549.10	746.67	5.42	24.96	234.19	26.94
3	825.08	1125.32	5.50	35.72	324.96	34.77
4	1103.03	1496.51	5.59	46.60	416.34	43.02
5	1381.13	1875.18	5.64	56.24	504.91	50.78
6	1658.12	2251.41	5.69	65.66	598.73	58.67
7	1935.49	2626.78	5.75	75.05	690.18	66.53
8	2213.18	2992.47	5.82	84.52	769.33	74.48
9	2490.69	3363.33	5.91	93.95	857.99	82.17
10	2769.49	3741.66	5.97	103.49	939.17	89.95
11	3047.16	4119.73	6.03	113.58	1003.28	97.73
12	3326.19	4510.99	6.12	124.38	1066.89	105.49
13	3606.64	4898.14	6.19	134.31	1066.89	113.23
14	3891.31	5275.63	6.25	145.94	1200.43	121.21
15	4178.92	5659.15	6.34	157.10	1266.57	131.12

data points and got 55% accurate results within less time because the Orkut dataset was vast and higher than other datasets. For less time, the reason is the K nearest neighbors are close to each other. The live-Journal dataset contained 3,997,963 data points and presented 76% accurate results on the value of $K = (1 \quad to \quad 15)$. A live-journal dataset is very high and broad and provides many effective results within less time than other datasets. However, the KD tee algorithm's overall performance continuously attains the six datasets' highest accurate results at the shortest computation time.

4.4.3 Results on refinement (NN-Descent) of KNN graph construction

Tables 4 and 5 show the NN-descent algorithm calculated accuracy and time of Facebook, Twitter, Live-Journal, YouTube, Orkut, and SIFT1M datasets from Eq. 16.

Figures 11 and 12 show the NN-Descent algorithm performance of the KNN graph w.r.t accuracy and time cost KNN graph. In Tables 4 and 5, we calculated the highly accurate results of the social network's datasets from the NN-descent algorithm. The Twitter dataset is complicated to construct the KNN graph at 95% accurate results using the NN-descent because it is so larger than the Facebook dataset. The SIFT1M dataset gets 100% accurate results in less time after the refinement technique. YouTube data present 45% accurate results for the NN-Descent algorithm because the YouTube dataset is broader than another dataset. However, after applying the NN-Descent algorithm, YouTube dataset accuracy decreases instead of increasing because the K nearest neighbors are far from each other. The live-Journal dataset shows the same accuracy, but the time is changed for the refinement algorithm. The Orkut dataset gets 100% accurate results within less time after applying the NN-descent algorithm. However, the overall performance of the KNN graph construction is continuously reaching 100% accurate results with the shortest computation time. The existing method has frequently achieved 100% results in both algorithms, and it can easily transform high-dimensional data into low-dimensional data.

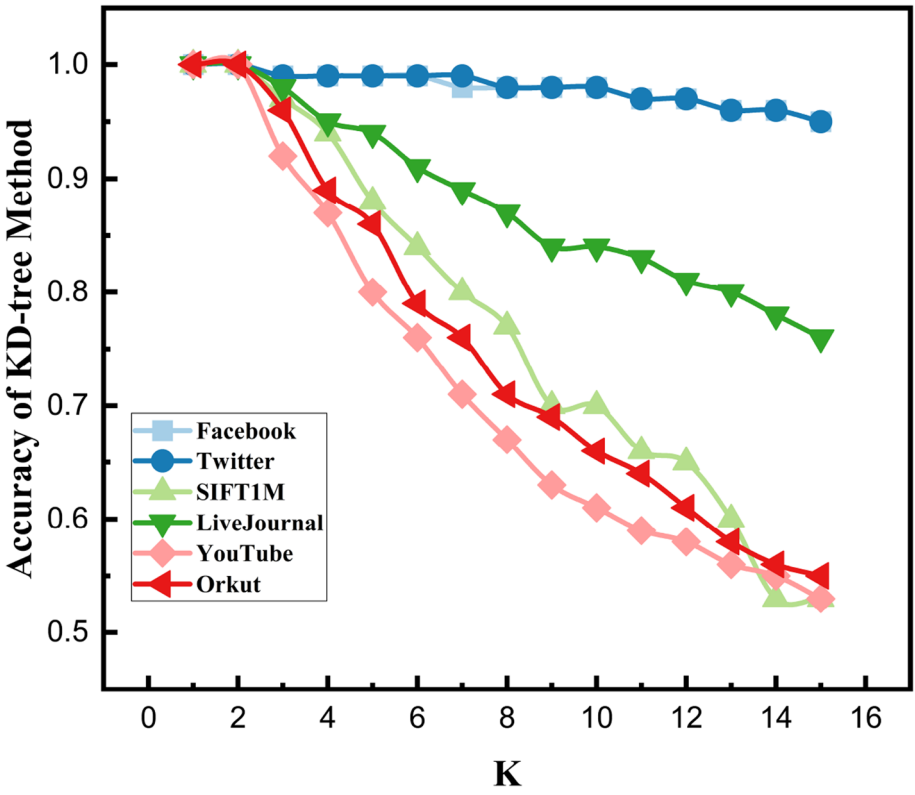

Fig. 9 Accuracy comparisons between Facebook, Twitter, SIFT1M, Live-Journal, YouTube, and Orkut Datasets

4.4.4 KD tree accuracy comparison between FastIsomap and Isomap algorithms

Figs. 13 and 14 compare the overall accuracy performance of the existing FastIsomap and Isomap with six datasets such as Facebook, Twitter, SIFT1M, YouTube, Live-Journal, and Orkut. The FastIsomap has generated 100% to 53% accurate results than the Isomap method in Tables 6 and 7. The Isomap algorithm is very time-consuming and slow for large and high-dimensional datasets.

For the Facebook dataset, the Isomap algorithm provided 86% to 48% accurate results. Building the KNN graph with 86% to 49% accurate results is tough through the Isomap algorithm on the Twitter dataset. For SIFT1M and YouTube, datasets achieved 76% to 40% accuracy on $K's$ value (1 to 15) via the Isomap algorithm. The Live-Journal dataset is presented with 89% to 50% accurate results through Isomap. The Isomap algorithm provides 79% to 38% accuracy for the Orkut dataset because Orkut is a vast dataset rather than other datasets. Therefore, Orkut dataset accuracy is decreased for the Isomap algorithm. Moreover, the existing method has produced very effective results for constructing the KNN graph, and the existing FastIsomap is much more significant and efficient than the Isomap algorithm.

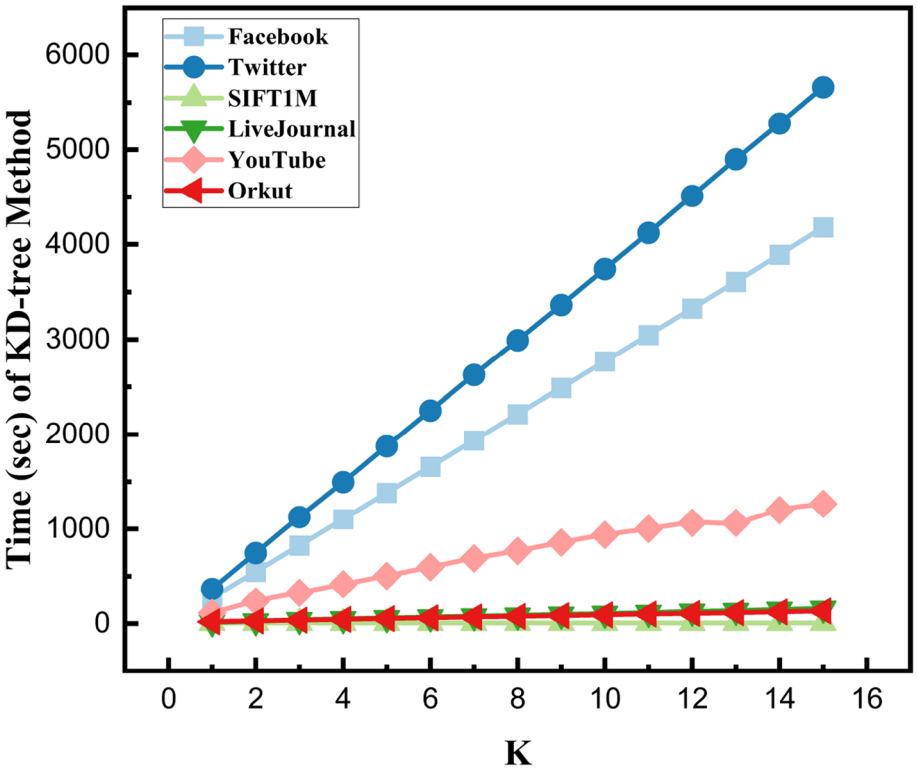

Fig. 10 Time (sec) comparisons between Facebook, Twitter, SIFT1M, Live-Journal, YouTube, and Orkut Datasets

Table 4 Comparison accuracy of NN-Descent of Facebook, Twitter, SIFT1M, Live-Journal, YouTube, and Orkut Datasets

K	Facebook	Twitter	SIFT1M	Live-Journal	YouTube	Orkut
1	1.00	1.00	1.00	1.00	1.00	1.00
2	1.00	1.00	1.00	1.00	1.00	1.00
3	1.00	1.00	1.00	0.98	0.90	1.00
4	1.00	1.00	1.00	0.95	0.83	1.00
5	1.00	1.00	1.00	0.94	0.76	1.00
6	1.00	0.99	1.00	0.91	0.71	1.00
7	1.00	0.99	1.00	0.89	0.66	1.00
8	1.00	0.98	1.00	0.87	0.62	1.00
9	1.00	0.98	1.00	0.85	0.58	1.00
10	1.00	0.98	1.00	0.84	0.56	1.00
11	1.00	0.97	1.00	0.83	0.53	1.00
12	1.00	0.97	1.00	0.81	0.51	1.00
13	1.00	0.96	1.00	0.80	0.49	1.00
14	1.00	0.96	1.00	0.78	0.47	1.00
15	1.00	0.95	1.00	0.76	0.45	1.00

Table 5 Time Comparison of NN-Descent of Facebook, Twitter, SIFT1M, Live-Journal, YouTube, and Orkut Datasets

K	Facebook	Twitter	SIFT1M	Live-Journal	YouTube	Orkut
1	72.71	905.58	4.88	9.54	6887.44	0.09
2	156.12	1988.83	5.31	18.76	13667.03	0.13
3	245.02	2976.66	5.66	27.89	20664.03	0.17
4	334.68	3815.51	6.00	37.09	27406.31	0.22
5	422.37	4647.37	6.34	46.23	34133.02	0.27
6	518.01	5440.17	16.34	55.76	40895.39	0.40
7	606.32	6104.28	16.69	64.90	48530.41	0.45
8	695.07	6723.18	17.04	74.07	56053.35	0.49
9	786.14	7342.48	17.38	83.20	63180.93	0.54
10	876.86	7972.88	17.73	92.33	69907.25	0.59
11	968.25	8588.63	18.11	101.44	76628.92	0.67
12	1059.77	9209.83	18.46	110.58	83328.51	0.74
13	1151.11	9815.17	18.81	119.67	90045.50	0.79
14	1241.73	10426.42	19.17	128.81	96734.74	0.84
15	1332.76	11041.22	19.54	137.94	103434.74	0.89

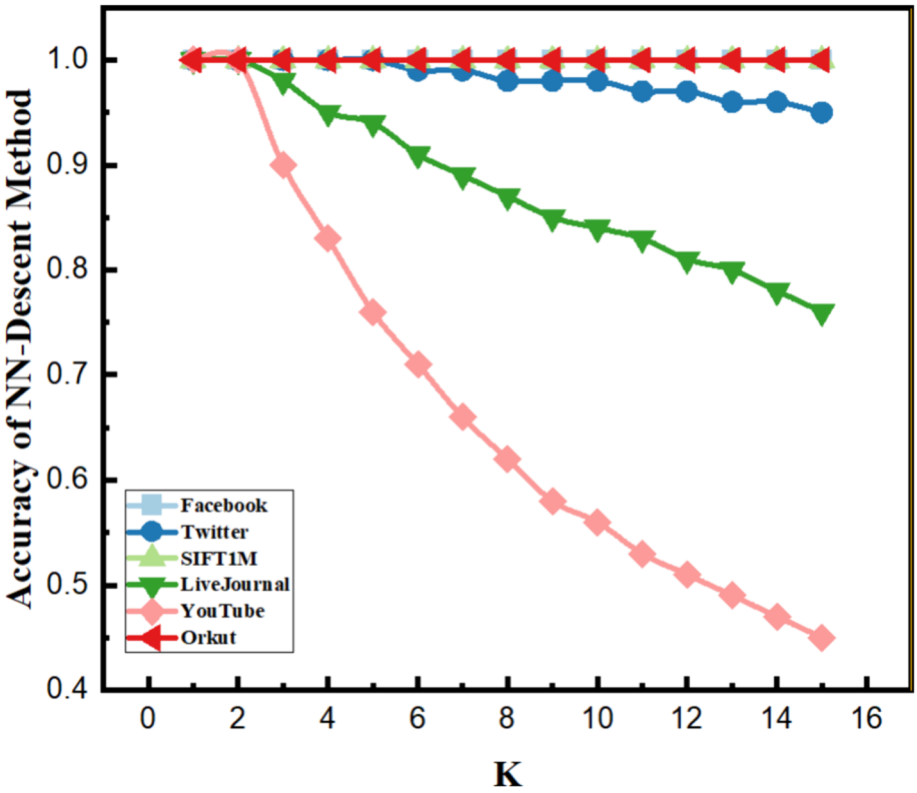

Fig. 11 Accuracy comparisons between Facebook, Twitter, SIFT1M, Live-Journal, YouTube, and Orkut Datasets

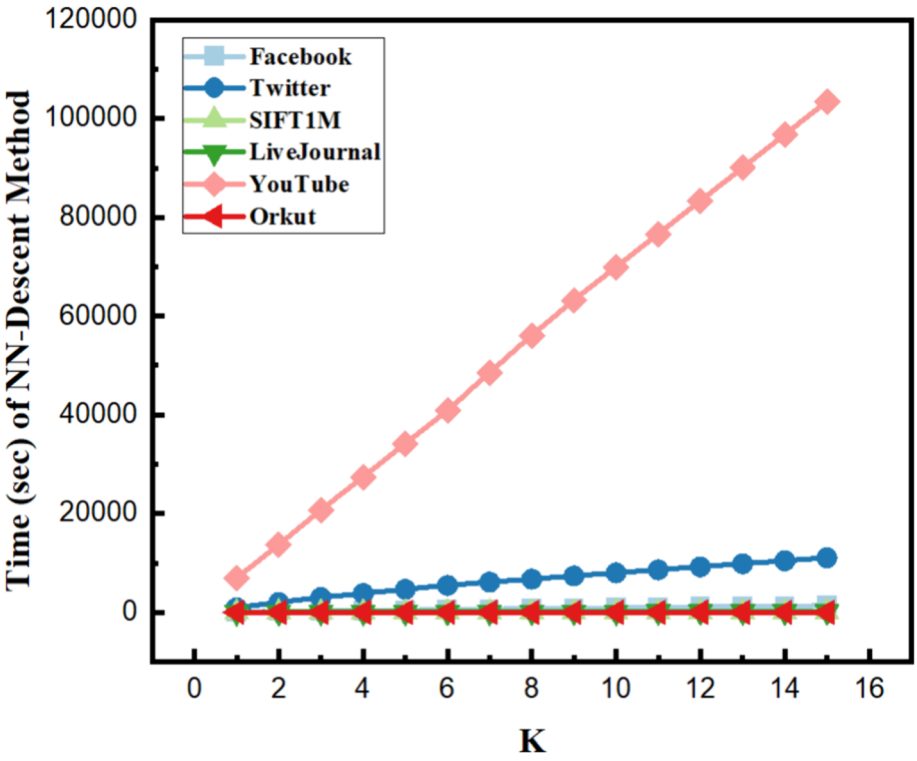

Fig. 12 Time (sec) comparison between Facebook, Twitter, SIFT1M, Live-Journal, YouTube, and Orkut Datasets

4.4.5 NN-descent accuracy comparison between existing fastisomap and isomap algorithms

Figures 15 and 16 show the overall accuracy comparison between existing FastIsomap and Isomap methods with the NN-descent algorithm. We calculated the social network's dataset's highly accurate results from the FastIsomap in Tables 8 and 9. The FastIsomap has provided 100% accurate results of Facebook, SIFT1M, and Orkut results compared to the Isomap on the value of $K = (1 \quad to \quad 15)$. However, the Isomap algorithm accuracy is the same for all datasets because the NN-Descent (refinement) algorithm increases the accuracy of our method. Our existing FastIsomap has produced constructive results in millions of datasets with hundreds of dimensions and constructed an accurate KNN graph. Moreover, it is much more significant and efficient than the Isomap algorithm.

5 FastIsomapVis with divide, conquer, and combine approach

This section of the survey paper addresses two challenges the Isomap algorithm encounters: topological stability and shortest path problems. Increasing the size of the K neighborhood can cause a rise in the geodesic distance, leading to a longer computation time for the shortest path data points. Additionally, geodesic distance is expensive and may not always provide reliable results, unlike the Euclidean distance measure E. Euclidean distance E for high-

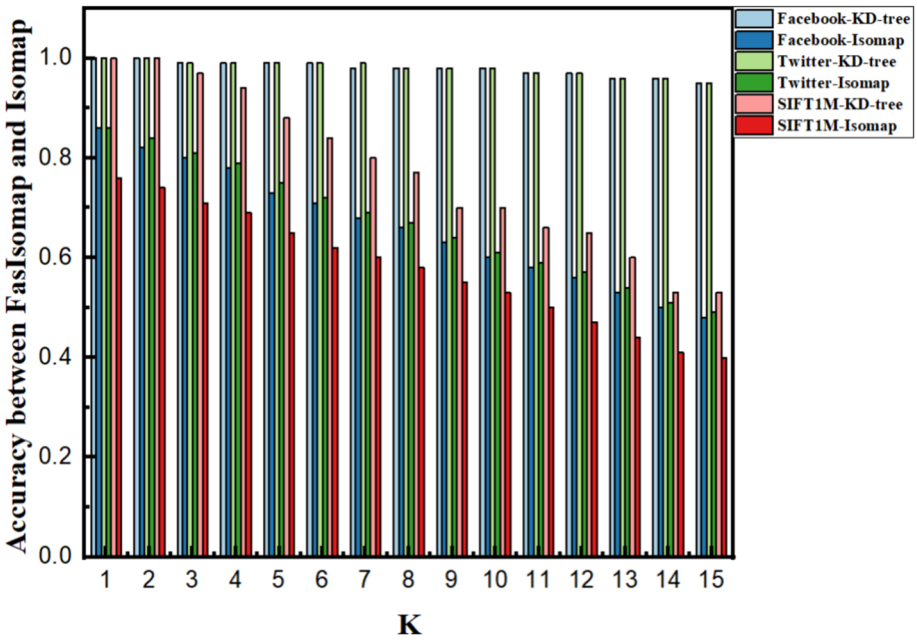

Fig. 13 Overall accuracy comparison between existing FastIsomap and Isomap algorithms with YouTube, Live-Journal, and Orkut datasets

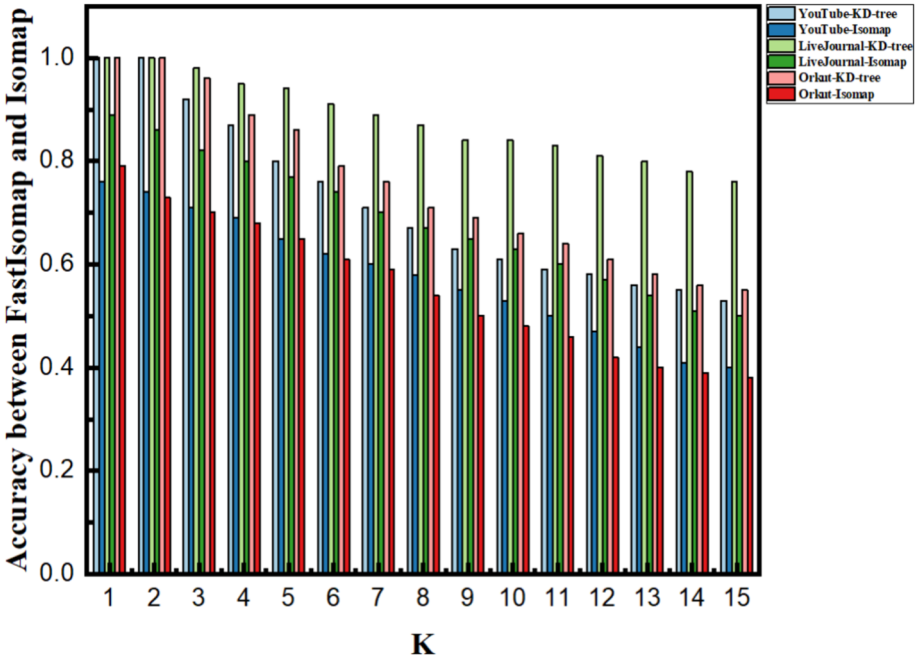

Fig. 14 Overall Accuracy Comparison between existing FastIsomap and Isomap Algorithms with YouTube, Live-Journal, and Orkut datasets

Table 6 KD tree Accuracy Comparison between existing FastIsomap and Isomap Algorithms of the Facebook, Twitter, and SIFT1M Datasets

Facebook			Twitter		SIFT1M	
K	FastIsomap	Isomap	FastIsomap	Isomap	FastIsomap	Isomap
1	1.00	0.86	1.00	0.86	1.00	0.76
2	1.00	0.82	1.00	0.84	1.00	0.74
3	0.99	0.80	0.99	0.81	0.97	0.71
4	0.99	0.78	0.99	0.79	0.94	0.69
5	0.99	0.73	0.99	0.75	0.88	0.65
6	0.99	0.71	0.99	0.72	0.84	0.62
7	0.98	0.68	0.99	0.69	0.80	0.60
8	0.98	0.66	0.98	0.67	0.77	0.58
9	0.98	0.63	0.98	0.64	0.70	0.55
10	0.98	0.60	0.98	0.61	0.70	0.53
11	0.97	0.58	0.97	0.59	0.66	0.50
12	0.97	0.56	0.97	0.57	0.65	0.47
13	0.96	0.53	0.96	0.54	0.60	0.44
14	0.96	0.50	0.96	0.51	0.53	0.41
15	0.95	0.48	0.95	0.49	0.53	0.40

dimensional datasets provides accurate results at a lower cost. The existing method used the Euclidean distance matrix $dis = \sqrt{x_i, x_j}$ instead of geodesic distances to address these issues. This enables us to calculate the dataset's similarities through a square matrix, which yields more cost-effective and accurate results. Moreover, we used the Dijkstra Buckets Double (DKD) algorithm to solve the shortest path problem and the KD tree algorithm to

Table 7 KD tree Accuracy Comparison between existing FastIsomap and Isomap Algorithms of the YouTube, Live-Journal, and Orkut Datasets

YouTube			Live-Journal		Orkut	
K	FastIsomap	Isomap	FastIsomap	Isomap	FastIsomap	Isomap
1	1.00	0.76	1.00	0.89	1.00	0.79
2	1.00	0.74	1.00	0.86	1.00	0.73
3	0.92	0.71	0.98	0.82	0.96	0.70
4	0.87	0.69	0.95	0.80	0.89	0.68
5	0.80	0.65	0.94	0.77	0.86	0.65
6	0.76	0.62	0.91	0.74	0.79	0.61
7	0.71	0.60	0.89	0.70	0.76	0.59
8	0.67	0.58	0.87	0.67	0.71	0.54
9	0.63	0.55	0.84	0.65	0.69	0.50
10	0.61	0.53	0.84	0.63	0.66	0.48
11	0.59	0.50	0.83	0.60	0.64	0.46
12	0.58	0.47	0.81	0.57	0.61	0.42
13	0.56	0.44	0.80	0.54	0.58	0.40
14	0.55	0.41	0.78	0.51	0.56	0.39
15	0.53	0.40	0.76	0.50	0.55	0.38

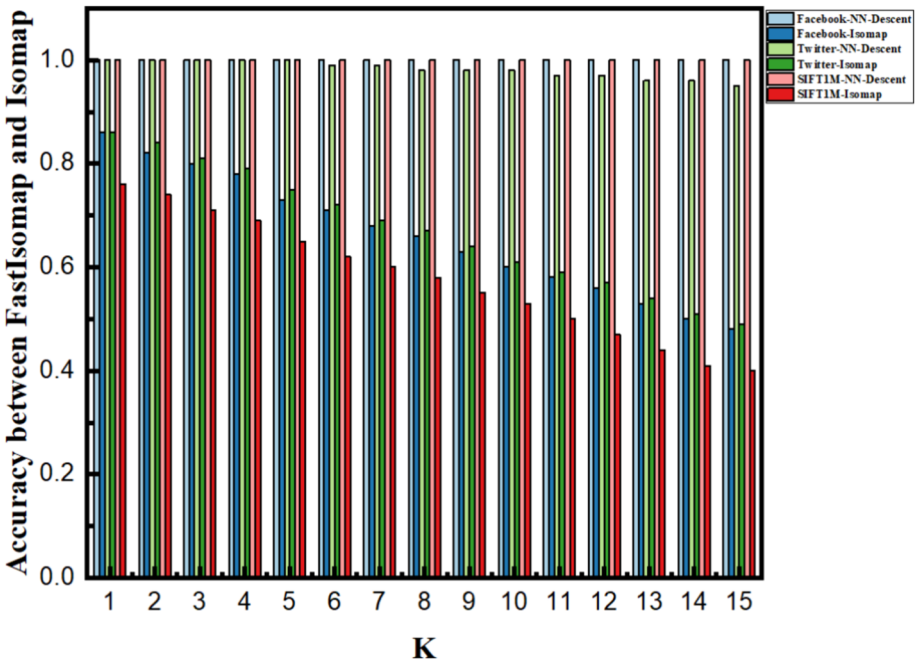

Fig. 15 Overall accuracy comparison between existing FastIsomap and Isomap algorithms of the Facebook, Twitter, and SIFT1M datasets

address the topological stability problem. We adopted a hierarchical approach that involves dividing, conquering, and combining to tackle both challenges of the Isomap algorithm. Figure 17 depicts the basic structure of our existing FastIsomapVis [48] method.

5.1 Hierarchical divide, conquer, and combine approach

The divide, conquer, and combine is a beneficial approach to reduce the time cost of the algorithm. The hierarchical approach recursively divides the problem into sub-problems, split into small issues that can quickly solve problems. After that, a combined approach can generate the results of the sub-problems of the actual problem. The hierarchical approach is better and easier for constructing the KNN graph G.

5.2 Divide approach

In the 'divide' step, we focus on the topological stability problem of the Isomap algorithm, and it solves this problem via the KD tree algorithm. The hierarchical divide approach is used to initialize the KNN graph data points better. So, we deal with the KNN case in a divided phase and split the KNN data points into sub-KNN data points. Alternatively, we have calculated the similarities of KNN data points through Euclidean distance E. The basic idea of the KD tree is to provide better initialization to improve the performance of Isomap significantly. See the detail of the KD tree algorithm in section 3 because we have used the same algorithm with the NN-descent algorithm in section 3 for the topological stability problem. That's why we have not mentioned the details here.

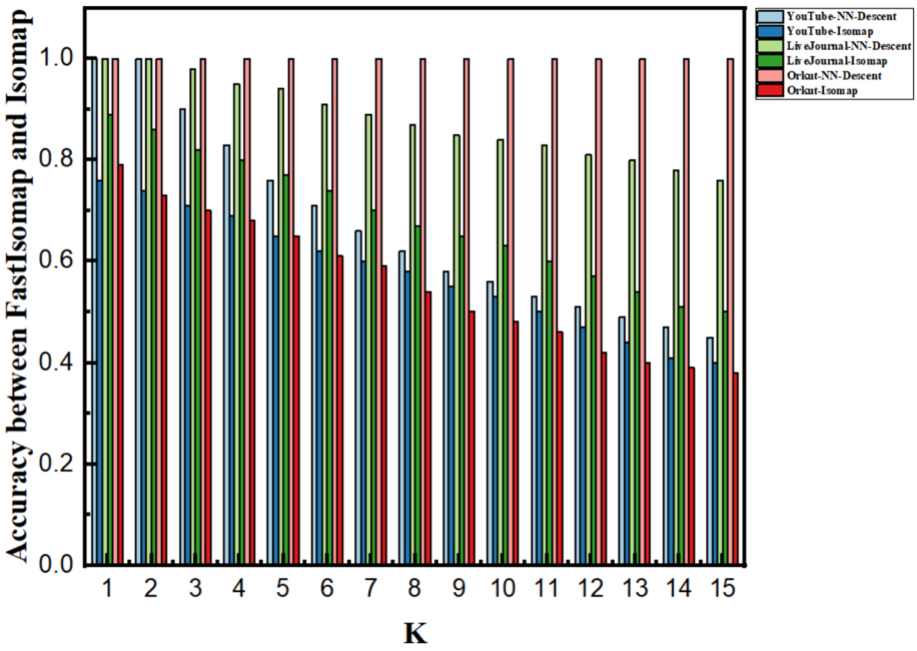

Fig. 16 Accuracy comparison between existing fastisomap and isomap algorithms of the YouTube, LiveJournal, and Orkut datasets

5.3 Conquer approach

In the 'Conquer' step of the Isomap algorithm, the primary focus is solving the shortest path problem. This problem is currently solved via the DKD algorithm, which calculates the shortest path for KNN data points and merges sub-KNN data points. It used the Euclidean distance E measure instead of the geodesic distance. It aims to enhance the 'Conquer' approach by reducing the number of data points utilizing the closest KNN data points for conquering and maintaining high-quality data points at each level. By implementing these enhancements, we believe that the efficiency and effectiveness of the Isomap algorithm can be improved.

5.3.1 General shortest path distance

The graph's mathematical concept is based on the vertices V and edges E in graph theory. The edges E is connected to more than one vertex, and edges have a numeric cost called weights. The shortest path distance means "finding a path between two vertices through minimum weights cost." In past years, various shortest-path distance algorithms were introduced. The shortest path distance algorithms are categorized into three forms: Single Source shortest path, Single Destination shortest path, and All-Pairs shortest path algorithms [74]. The classification of the shortest path distance algorithms is shown in Fig. 18.

5.3.2 Description of general shortest path distance

In Table 10 [75], we briefly define the details of 17 General Shortest-Path Algorithms (GSPA). The Dijkstra algorithm is used in numerous areas, such as Link-State Routing (LSR) protocols [76], Open Shortest Path First (OSPF) [77], Logistic Distribution Line [78] and Robot Path

Table 8 NN-Descent Accuracy Comparison between existing FastIsomap and Isomap algorithms of the Facebook, Twitter, and SIFT1M Datasets

Facebook			Twitter		SIFT1M	
K	FastIsomap	Isomap	FastIsomap	Isomap	FastIsomap	Isomap
1	1.00	0.86	1.00	0.86	1.00	0.76
2	1.00	0.82	1.00	0.84	1.00	0.74
3	1.00	0.80	1.00	0.81	1.00	0.71
4	1.00	0.78	1.00	0.79	1.00	0.69
5	1.00	0.73	1.00	0.75	1.00	0.65
6	1.00	0.71	0.99	0.72	1.00	0.62
7	1.00	0.68	0.99	0.69	1.00	0.60
8	1.00	0.66	0.98	0.67	1.00	0.58
9	1.00	0.63	0.98	0.64	1.00	0.55
10	1.00	0.60	0.98	0.61	1.00	0.53
11	1.00	0.58	0.97	0.59	1.00	0.50
12	1.00	0.56	0.97	0.57	1.00	0.47
13	1.00	0.53	0.96	0.54	1.00	0.44
14	1.00	0.50	0.96	0.51	1.00	0.41
15	1.00	0.48	0.95	0.49	1.00	0.40

Table 9 NN-descent accuracy comparison between existing FastIsomap and Isomap algorithms with YouTube, Live-Journal, and Orkut datasets

YouTube			Live-Journal		Orkut	
K	FastIsomap	Isomap	FastIsomap	Isomap	FastIsomap	Isomap
1	1.00	0.76	1.00	0.89	1.00	0.79
2	1.00	0.74	1.00	0.86	1.00	0.73
3	0.90	0.71	0.98	0.82	1.00	0.70
4	0.83	0.69	0.95	0.80	1.00	0.68
5	0.76	0.65	0.94	0.77	1.00	0.65
6	0.71	0.62	0.91	0.74	1.00	0.61
7	0.66	0.60	0.89	0.70	1.00	0.59
8	0.62	0.58	0.87	0.67	1.00	0.54
9	0.58	0.55	0.85	0.65	1.00	0.50
10	0.56	0.53	0.84	0.63	1.00	0.48
11	0.53	0.50	0.83	0.60	1.00	0.46
12	0.51	0.47	0.81	0.57	1.00	0.42
13	0.49	0.44	0.80	0.54	1.00	0.40
14	0.47	0.41	0.78	0.51	1.00	0.39
15	0.45	0.40	0.76	0.50	1.00	0.38

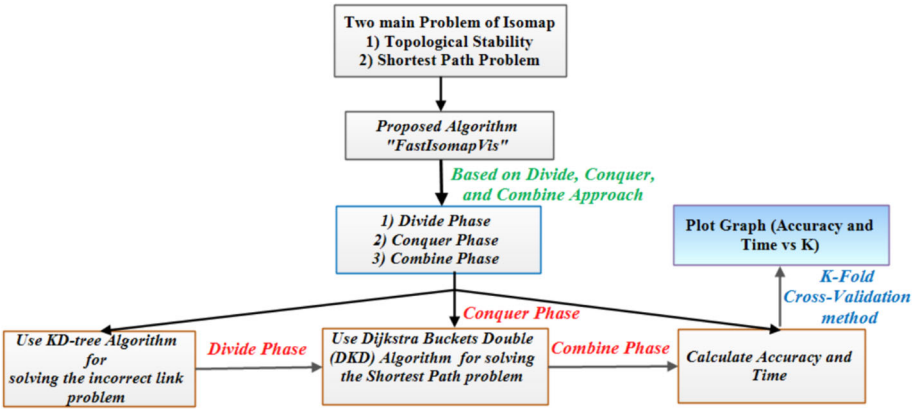

Fig. 17 Basic structure of the existing FastIsomapVis [48] method

Planning [79]. The previously proposed improved Isomap method [44, 45] is based on the Dijkstra algorithm with the Fibonacci Heap (Fib-Dij). The Fib-Dij algorithm can handle all pairs of shortest paths data points based on the neighborhood graph. This method is much faster than Dijkstra and Floyd's algorithms; it can improve speed and reduce the Isomap algorithm's time cost.

5.3.3 Dijkstra buckets double algorithm

The DKD algorithm implementation is based on two algorithms, including Dijkstra Bucket Overflow Bag (DKM) and Dijkstra Buckets Approximate (DKA), [75]. It contains random buckets managed by implementing the DKD and making an arbitrary level buckets matrix. According to Algorithm 5, we have computed the cost of the shortest path of the KNN and searched for the shortest path value of KNN through the Breath First Search (BFS) algorithm in lines (3 to 6). In line 7, we have sorted the path values of the KNN in random buckets via a bubble sort algorithm. In lines (8 to 18), we have identified the minimum path cost of the KNN and then created a matrix of the resulting path values of the KNN data points. The DKD is much faster than Dijkstra, and Floyd's algorithms present high KNN graph accuracy, reducing the Isomap algorithm's time cost.

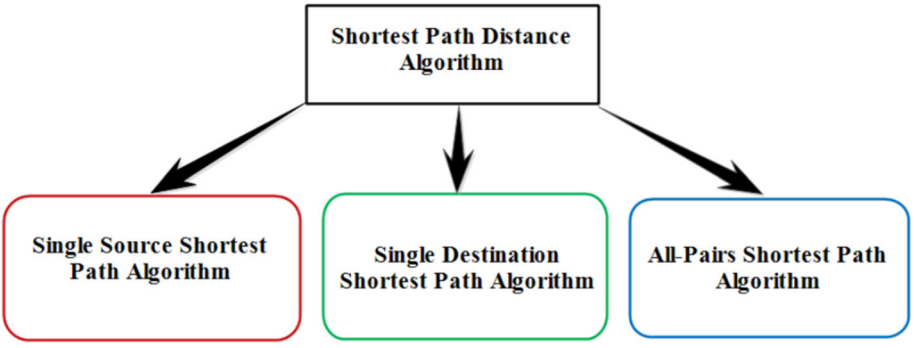

Fig. 18 Classification of shortest path algorithm

Table 10 17-common shortest path algorithms

Algorithm Names	Algorithm Abbreviations	Algorithm Implementations in Data Structures
Bellman-Ford-Moore	BFM	Bellman-Ford-Moore
	BFP	Bellman-Ford-Moore with parent- checking
Dijkstra	DKQ	Dijkstra's Naive Implementation
	DKB	Dijkstra's Buckets - Basic Implementation
	DKD	Dijkstra's Buckets - Double Implementation
	DKA	Dijkstra's Buckets - Approximate
	DKM	Dijkstra's Buckets - Overflow Bag
	DKF	Dijkstra's Heap - Fibonacci
	DKH	Dijkstra's Heap - k-Array
	DKR	Dijkstra's Heap - R-Heap
A*-Search	ASH	A*-Search algorithm with k-array Heap
	ASBD	A*-Search algorithm with Double Buckets
	ASBA	A*-Search algorithm with Approximate Buckets of data structures
Graph Growth Model	TQQ (Two-Q)	Graph Growth – Gallo and Pallottino algorithm with two queues data structures
	PAP	Graph Growth - Pape
Threshold Algorithm	THR	Threshold Algorithm
Topological Ordering	GR1	Topological Ordering - Basic
Topological Ordering	GR2	Topological Ordering - Distance Updates

Algorithm 5 Dijkstra buckets double algorithm of FastIsomapVis.

Require: Approximate KNN graph: K, No of nodes: N
Ensure: An Approximate KNN graph: G The Division step
1: Using Algorithm 2 to Construction-TREE function and R shows to the tree input
2: The Conquer step with DKD
3: **for** (K ∈ N) **do**
4: Colo=Cost(K)
5: ml= Colo(Find(Colo))
6: mat=Sort(UNIQUE(ml))
7: **for** (j ∈ length (mat))) **do**
8: mx= mat(j)
9: Temp=MIN (Cost(Colo).[ ],2)
10: Temp=Temp + mx
11: **end for**
12: Cost(K)=Colo
13: **for** (K ∈ N) **do**
14: Cost(K,K)=r
15: **end for**
16: **end for**
17: End Conquer step

5.4 Combine approach

In the 'combine' step, we prioritize the accuracy and efficiency of the FastIsomapVis [48] and Isomap algorithms. To evaluate the overall accuracy of the existing method through the KD

tree searcher algorithm, which facilitates the construction of the K-nearest neighbors (KNN) graph, denoted as G. For detailed information regarding the KD tree searcher method, please refer to Section 3, as we have employed the same algorithm to assess accuracy and time. Therefore, we have not provided a detailed explanation here. Moreover, the existing method primarily employs hierarchical divide, conquer, and combine approaches. These approaches leverage an efficient KD tree algorithm to generate the KNN graph with higher accuracy, merge sub-KNN graph data points, and enhance the graph data via the DKD algorithm.

5.5 Experimental results

Our experimental results evaluate the efficiency of the existing FastIsomapVis method on high and large-dimensional datasets. Implementing the existing FastIsomapVis method is based on the nine social network datasets, Facebook, Twitter, Amazon, Live-Journal, YouTube, and Orkut, available on [65]. 20Newsgroups [80], SIFT1M [66], and Perfume [81, 82]. The dataset's details are listed in Table 11.

5.5.1 Comparisons between accuracy and time of the existing FastIsomapVis algorithm

Tables 12 and 13 show the accuracy and time of Facebook, Twitter, 20Newsgroups, Amazon, SIFT1M, Live-Journal, YouTube, Orkut, and Perfume datasets from (16).

Figures 19 and 20 represent the graph of time and accuracy for nine high-dimensional datasets. The graphs of the existing FastIsomapVis algorithm consistently attains highly accurate results with less computation time. We have used $K's$ value from 1 to 15 to calculate the datasets' time and accuracy.

Facebook datasets consist of 4039 data points. In Fig. 19, we have calculated 100% accurate results on $K = 1$ to 6 values and got 0.99% results for $K's = 7$ to 15 values. In Fig. 20, the result of calculated time is increased linearly from 916.34 to 30488.28 seconds. The slight decrease in accuracy is high values of K because some neighboring data points are far from each other.

Twitter and Live-Journal consist of 81,306 and 3,997,963 data points. In Fig. 19, we observed that the accuracy is 100% for $K's$=1 to 4 initial values, and it gradually decreases with the increase of K=5 to 15 values. This strange behavior is that Twitter and Live-Journal datasets have large categories compared to their dimensions and data points, as shown in Table 11. In Table 13, the computation time of Live-Journal is less than Twitter because

Table 11 Properties of datasets

Datasets	Data	Dimensions	Categories
Facebook	4,039	40	10
Twitter	81,306	40	973
20Newsgroups	18,846	100	30
Amazon	334,863	100	1000
SIFT1M	100,000	128	1000
YouTube	1,134,890	150	5000
Orkut	3,072,441	150	5000
Live-Journal	3,997,963	100	5000
Perfume	560	40	20

Table 12 Accuracy Comparison between Facebook, Twitter, 20Newsgroups Amazon, SIFT1M, Live-Journal, YouTube, Orkut, and Perfume Datasets of the existing FastIsomapVis Method

K	Fabok	Twit	SI1M	Liv-Jou	YTube	Orkut	20Negps	Perfume	Amazon
1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
3	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
4	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
5	1.00	0.90	1.00	0.88	0.99	0.86	1.00	1.00	1.00
6	1.00	0.90	1.00	0.88	0.99	0.85	1.00	0.99	1.00
7	0.99	0.81	1.00	0.82	0.99	0.82	1.00	0.99	1.00
8	0.99	0.81	1.00	0.82	0.99	0.81	1.00	0.99	1.00
9	0.99	0.73	1.00	0.78	0.99	0.77	1.00	0.99	1.00
10	0.99	0.73	1.00	0.78	0.99	0.76	1.00	0.99	1.00
11	0.99	0.67	1.00	0.75	0.99	0.74	1.00	0.99	1.00
12	0.99	0.67	1.00	0.75	0.99	0.73	1.00	0.98	1.00
13	0.99	0.61	1.00	0.73	0.99	0.72	1.00	0.98	1.00
14	0.99	0.61	1.00	0.73	0.99	0.70	1.00	0.97	1.00
15	0.99	0.56	1.00	0.71	0.99	0.68	1.00	0.97	1.00

some neighboring data points are close to each other, and some neighboring data points are far away.

Tables 12 and 13 show the time and accuracy of SIFT1M, 20Newsgroups, and Amazon datasets. Figs 19 and 20 show that our proposed existing algorithm consistently attains highly accurate results in less computation time. For 20Newsgroups, SIFT1M, Amazon datasets consist of 18,306, 100,000, and 334,863 data points, respectively. In Fig. 19, we have achieved

Table 13 Time comparison between Facebook, Twitter, 20Newsgroups Amazon, SIFT1M, Live-Journal, YouTube, Orkut, and perfume datasets of the existing FastIsomapVis method

K	Fabok	Twit	SI1M	Liv-Jou	YTube	Orkut	20Negps	Perfume	Amazon
1	916.34	1962.87	1.56	104.07	76.45	309.91	4.09	0.11	5.06
2	1803.71	3928.44	2.11	203.41	192.26	578.71	6.68	0.17	5.41
3	2693.08	5954.39	2.66	304.33	381.08	842.30	9.12	0.23	5.73
4	3581.71	7914.29	3.21	403.93	692.89	1117.05	11.56	0.28	6.12
5	4471.05	9895.67	3.78	503.90	965.82	1405.75	14.05	0.35	6.45
6	5360.13	12114.59	4.34	603.39	1326.26	1680.49	17.35	0.42	6.77
7	6250.56	14384.83	4.91	703.08	1545.25	1958.04	21.68	0.48	9.05
8	7138.51	16351.28	5.49	806.13	1853.34	2247.46	24.99	0.53	9.39
9	8027.64	18320.05	6.06	938.82	2385.68	2533.73	28.07	0.58	9.73
10	8918.18	20289.11	6.64	1051.08	2738.47	2814.67	30.60	0.63	10.07
11	9808.45	22258.83	7.22	1169.39	3157.59	3142.99	33.09	0.69	10.42
12	10705.27	24228.92	7.81	1285.96	3469.92	3436.23	35.59	0.74	10.78
13	11597.22	26198.13	8.40	1409.72	3764.28	3702.96	38.44	0.80	11.12
14	12489.69	28170.59	9.00	1529.83	4109.18	3971.73	41.08	0.86	11.47
15	30488.28	30139.75	9.61	1657.03	5891.25	4239.76	43.59	0.92	11.81

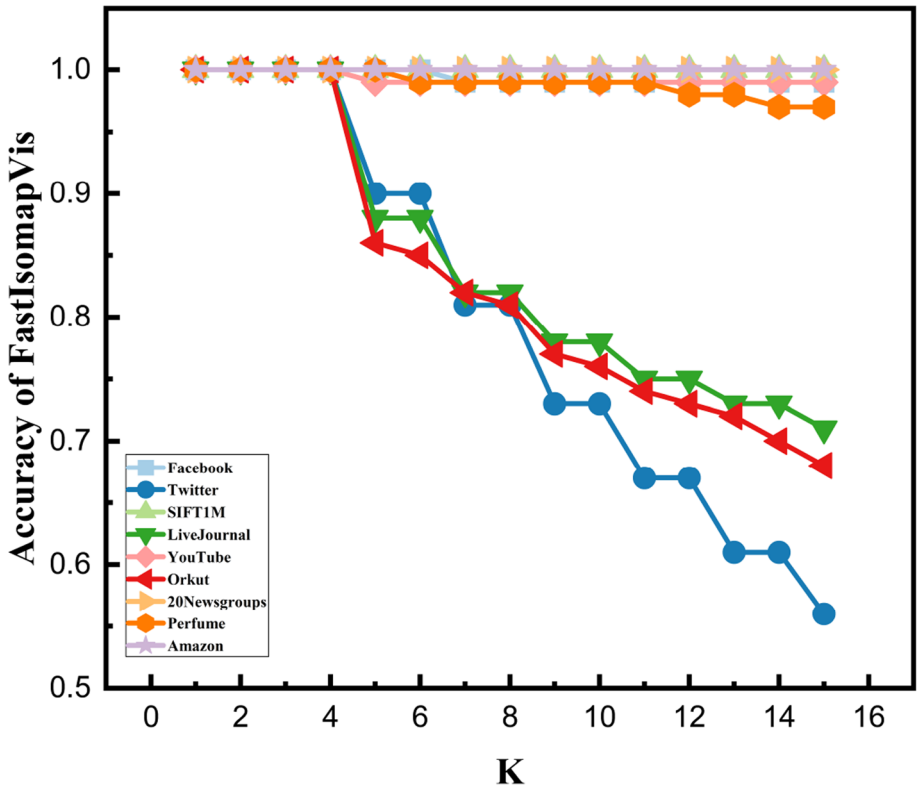

Fig. 19 Accuracy comparison between Facebook, Twitter, 20Newsgroups, Amazon, SIFT1M, Live-Journal, YouTube, Orkut, and perfume datasets

100% accurate results for all $K's$ values. The computation time to calculate these results increased linearly with $K's$ increase, as shown in Fig. 20.

The YouTube dataset consists of 1,134,890 data points. In Fig. 19, we have calculated 100% accurate results for $K's$=1 to 4 values and got 0.99% results for $K's$= 5 to 15 higher values. In Fig. 20, the calculated time results are increased linearly from 76.45 to 5891.25 seconds. The reason for a slight decrease in accuracy for higher values of K is that some neighboring data points are close to each other, and some data points are far away.

Orkut dataset consists of 3,072,441 data points. In Fig. 19, we have observed that the accuracy is 100% for K=1 to 4 initial values, but it gradually decreases with $K's$ increase 5 to 15 values. Table 11 shows that the strange behavior of the Orkut dataset has large categories compared to YouTube and Perfume, their dimensions, and data points.

Perfume datasets consist of 560 data points. In Fig. 19, we calculated 100% accurate results for values of K=1 to 5 values and got 0.97% results for higher values of K=6 to 15. The computation time to calculate these results increased linearly with K, as shown in Fig. 20.

In Fig. 19, the three datasets have 100% accuracies; therefore, the graph data points overlap. Moreover, Facebook and Twitter datasets have the same computation time values; this strange behavior of Twitter datasets is that neighboring data points and directed graphs are close.

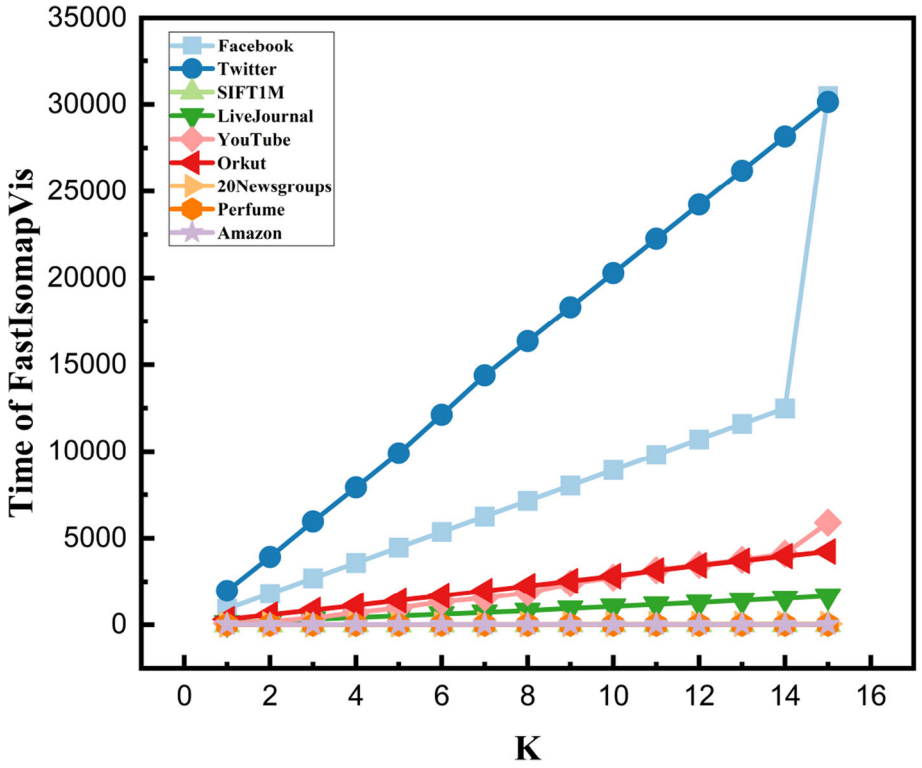

Fig. 20 Time (sec) Comparison between Facebook, Twitter, 20Newsgroups, Amazon, SIFT1M, Live-Journal, YouTube, Orkut, and Perfume datasets

5.5.2 Accuracy comparisons between existing FastIsomapVis and Isomap algorithms

In Figs. 21, 22 and 23, we have compared the overall accuracy of the nine different datasets on existing FastIsomapVis and Isomap methods. We computed highly accurate results from three datasets, including 20Newsgroups, Amazon, and SIFT1M, using the FastIsomapVis method. FastIsomapVis is provided with 100% accuracy compared to Isomap, whereas the Isomap algorithm is computationally expensive for large and high-dimensional datasets. In Tables 14 and 16, it is challenging to construct the KNN graph with 89% to 29% accuracy through the Isomap algorithm on K (1 to 15) with Facebook, Twitter, Live-Journal, YouTube, Orkut, and Perfume datasets. It is also evident that the accuracy of Isomap decreases significantly with the increase of K values. In contrast, the FastIsomapVis method has more stable and accurate results than the Isomap. Tables 14, 15 and 16 and Figs 21 to 23 specify that the existing FastIsomapVis is much more accurate and less sensitive to K values than the Isomap.

6 Noise removal isomap (NR-Isomap) method

This section introduces a novel method called NR-Isomap, which addresses the issue of noise-induced topological instability in the Isomap algorithm. Given a data point N in the l-

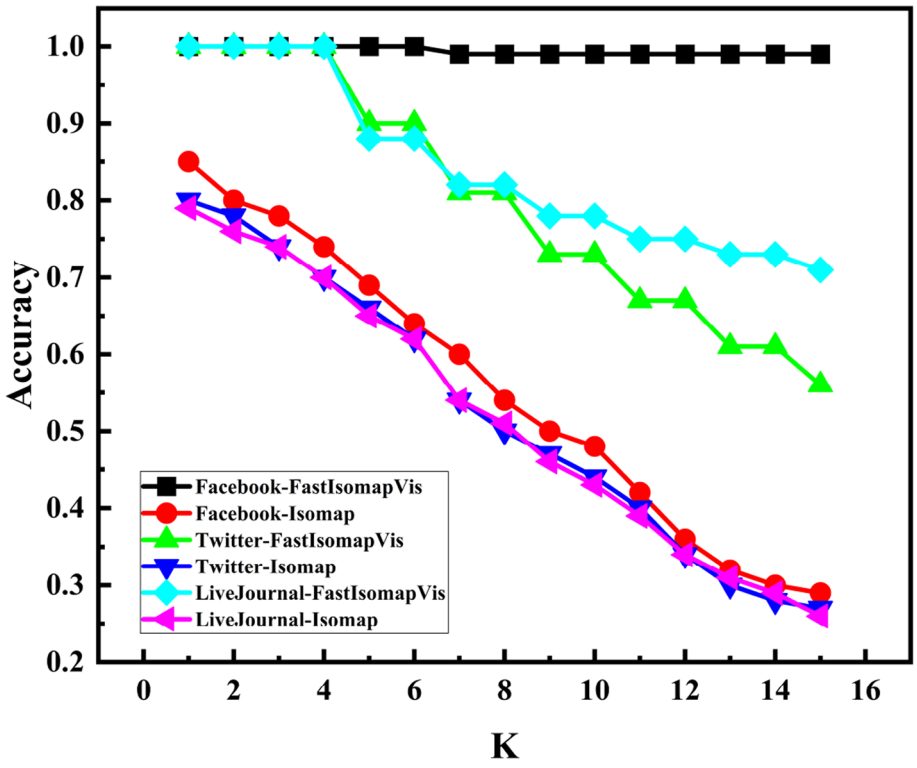

Fig. 21 Overall comparison accuracy between existing FastIsomapVis and Isomap Methods

dimensional low-dimensional data, where l represents the number of dimensions, we denote each data point X_i with an l-dimensional vector $Xi \in R^l (i = 1, 2, \ldots, N)$. Our previous method aims to identify and remove the noisy data points, allowing for embedding noise-free data points into a low-dimensional space. Fig. 24 illustrates the main steps involved in the existing NR-Isomap method, which are as follows:

1. Firstly, identify the noise from data points and then perform the LTSA algorithm for removing the noise in the original datasets.
2. We have used the Isomap algorithm to clear the data points.
3. We apply the Dijkstra algorithm to estimate the shortest path distance between the data points.
4. In the end, we use the MDS algorithm to map the data into low-dimensional embedded data after de-noising.

6.1 Short circuit edges elimination method

Tenenbaum et al. [6] identified a topological instability problem in the Isomap algorithm, which arises from selecting the nearest neighborhood size for constructing the neighborhood graph. This problem manifests when using a large neighborhood size, leading to the

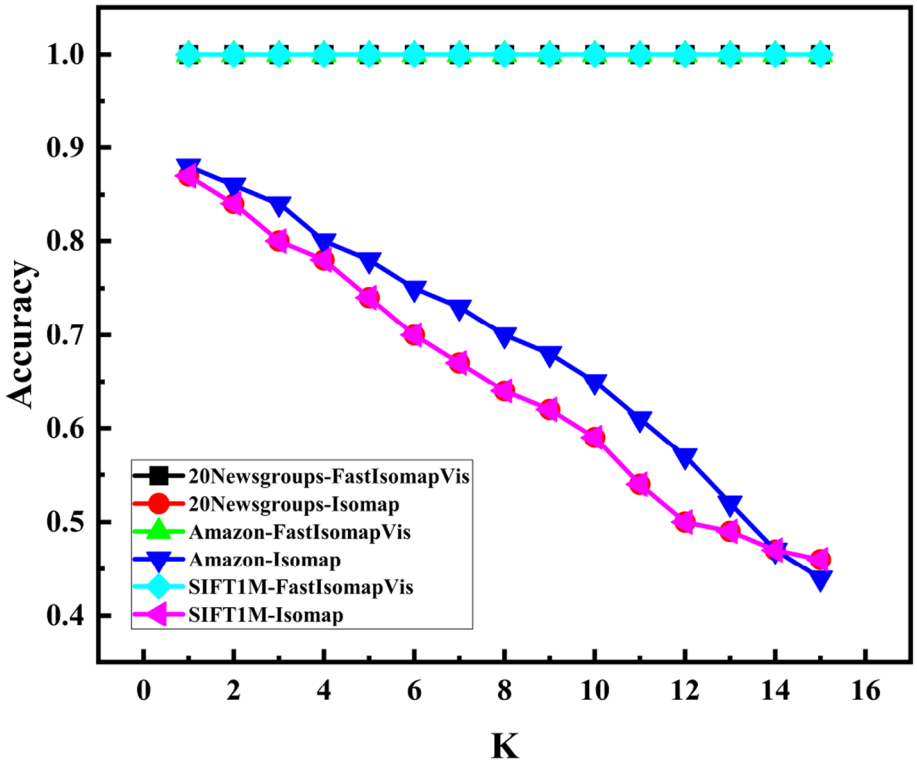

Fig. 22 Overall comparison accuracy between existing FastIsomapVis and Isomap methods

emergence of short-circuit edges that disrupt the underlying manifold geometry of the data points. Conversely, choosing a small neighborhood size becomes challenging, particularly in the presence of noisy data. This critical issue, often called the Topological Instability Problem (TIP), directly connects short-circuit edges between two sub-manifold data points. Furthermore, excessively reducing the neighborhood size can result in disconnected manifolds, a crucial concern in LLE [29]. To address these challenges and mitigate noise-induced topological instability, we propose the Noise Removal Isomap (NR-Isomap) algorithm as a solution for the Isomap algorithm.

6.2 Local Tangent Space Alignment (LTSA)

Zhenyue et al. introduced the dimensionality reduction embedding method called the LTSA method [52]. It can efficiently preserve a nonlinear embedding into low-dimensional datasets from high-dimensional datasets. LTSA can also rebuild the high-dimensional coordinates from embedding coordinates. This method is based on similar geometric intuitions as LLE; if a dataset is getting from a local linear manifold, the nearest neighbors of each continuous dataset are close and collected in the low-dimensional data. In LLE, every dataset point is linearly embedded into a locally linear patch of the manifold and then constructed in the low-dimensional data; consequently, the original dataset's locally linear relation is preserved. The LTSA method simultaneously creates a locally linear patch using a PCA method on the neighbors. Then, the patch can be evaluated as a local linear tangent space. Because the

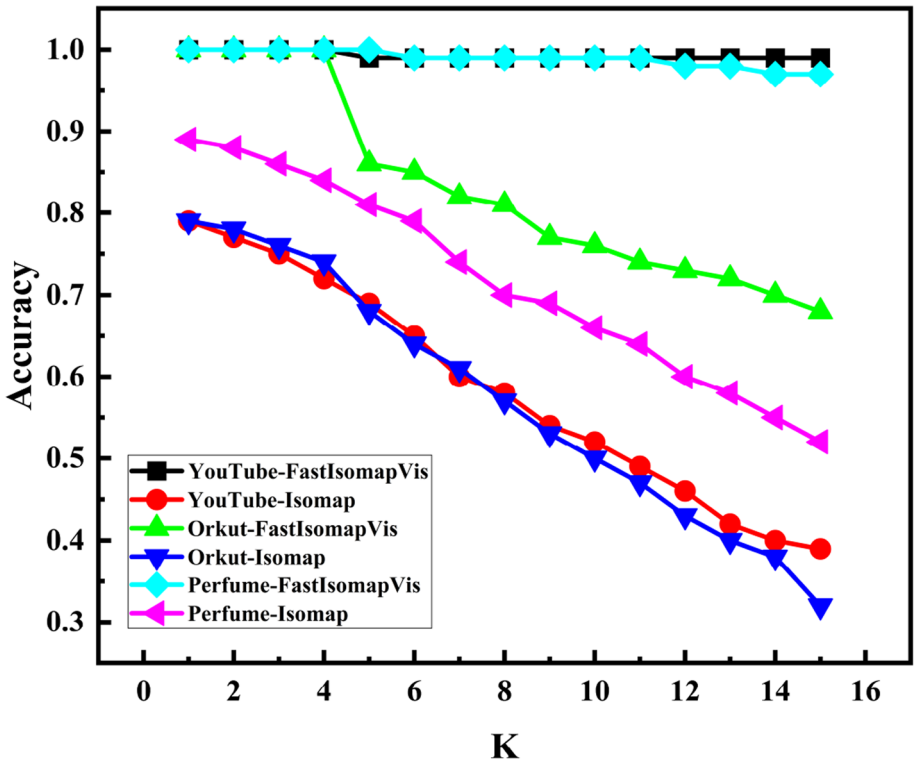

Fig. 23 Overall comparison accuracy between existing FastIsomapVis and Isomap methods

tangent location provides coordinate neighbors representation, the coordinates' neighbors provide a low-dimensional patch representation [83]. The LTSA Algorithm 6 is given below:

6.3 Dijkstra algorithm

In graph theory, the Dijkstra algorithm can be used for all-pairs and single-source shortest-path problems. The Dijkstra algorithm finds the path through the minimum path value between the nod and every node for the given node. Isomap calculates the geodesic distance between all nearest neighborhood data by estimating the shortest path distance using Dijkstra and Floyd's algorithm. Also, generate the resultant matrix's shortest path distance. If X_i and X_j are K nearest neighbor data points, the geodesic distance can be estimated by the shortest path $min[D_G(X_i, X_j)]$ distance between X_i and X_j is the neighborhood graph G using (17) [36, 84].

$$D_G(X_i, X_j) = min[D_G(X_i, X_j), D_G(X_i, X_n) + D_G(X_n, X_j)] \tag{17}$$

6.4 Multidimensional Scaling (MDS)

MDS contains various techniques which can be used to reduce the dimensionality of the data points. These techniques are used in numerous research fields, such as Data Mining,

Table 14 Comparison accuracy between existing FastIsomapVis and Isomap Methods of the Facebook, Twitter, and Live-Journal Datasets

Facebook			Twitter		Live-Journal	
K	FastIsoVis	Isomap	FastIsoVis	Isomap	FastIsoVis	Isomap
1	1.00	0.85	1.00	0.80	1.00	0.79
2	1.00	0.80	1.00	0.78	1.00	0.76
3	1.00	0.78	1.00	0.74	1.00	0.74
4	1.00	0.74	1.00	0.70	1.00	0.70
5	1.00	0.69	0.90	0.66	0.88	0.65
6	1.00	0.64	0.90	0.62	0.88	0.62
7	0.99	0.60	0.81	0.54	0.82	0.54
8	0.99	0.54	0.81	0.50	0.82	0.51
9	0.99	0.50	0.73	0.47	0.78	0.46
10	0.99	0.48	0.73	0.44	0.78	0.43
11	0.99	0.42	0.67	0.40	0.75	0.39
12	0.99	0.36	0.67	0.34	0.75	0.34
13	0.99	0.32	0.61	0.30	0.73	0.31
14	0.99	0.30	0.61	0.28	0.73	0.29
15	0.99	0.29	0.56	0.27	0.71	0.26

Table 15 Comparison accuracy between existing FastIsomapVis and Isomap of the 2NewsGroup, SIF1M and Amazon Datasets

20Newsgroups			Amazon		SIFT1M	
K	FastIsoVis	Isomap	FastIsoVis	Isomap	FastIsoVis	Isomap
1	1.00	0.87	1.00	0.88	1.00	0.87
2	1.00	0.84	1.00	0.86	1.00	0.84
3	1.00	0.80	1.00	0.84	1.00	0.80
4	1.00	0.78	1.00	0.80	1.00	0.78
5	1.00	0.74	1.00	0.78	1.00	0.74
6	1.00	0.70	1.00	0.75	1.00	0.70
7	1.00	0.67	1.00	0.73	1.00	0.67
8	1.00	0.64	1.00	0.70	1.00	0.64
9	1.00	0.62	1.00	0.68	1.00	0.62
10	1.00	0.59	1.00	0.65	1.00	0.59
11	1.00	0.54	1.00	0.61	1.00	0.54
12	1.00	0.50	1.00	0.57	1.00	0.50
13	1.00	0.49	1.00	0.52	1.00	0.49
14	1.00	0.47	1.00	0.47	1.00	0.47
15	1.00	0.46	1.00	0.44	1.00	0.46

Table 16 Comparison accuracy between existing FastIsomapVis and Isomap of the YouTube, Orkut, and Perfume Datasets

YouTube			Orkut		Perfume	
K	FastIsoVis	Isomap	FastIsoVis	Isomap	FastIsoVis	Isomap
1	1.00	0.79	1.00	0.79	1.00	0.89
2	1.00	0.77	1.00	0.78	1.00	0.88
3	1.00	0.75	1.00	0.76	1.00	0.86
4	1.00	0.72	1.00	0.74	1.00	0.84
5	0.99	0.69	0.86	0.68	1.00	0.81
6	0.99	0.65	0.85	0.64	0.99	0.79
7	0.99	0.60	0.82	0.61	0.99	0.74
8	0.99	0.58	0.81	0.57	0.99	0.70
9	0.99	0.54	0.77	0.53	0.99	0.69
10	0.99	0.52	0.76	0.50	0.99	0.66
11	0.99	0.49	0.74	0.47	0.99	0.64
12	0.99	0.46	0.73	0.43	0.98	0.60
13	0.99	0.42	0.72	0.40	0.98	0.58
14	0.99	0.40	0.70	0.38	0.97	0.55
15	0.99	0.39	0.68	0.32	0.97	0.52

Cryptography, Multi-objective Optimization, Psychology, and Marketing. It can map the data points through Euclidean distance, suitable for finding global geometry structure and optimal data representation [84]. Isomap is a variant of the MDS to model nonlinear data points using its geodesic distance. It estimates the low-dimensional coordinates using a distance matrix $D_G(X_i, X_j)$ and maps data into low-dimensional embedded data. It preserves as much as possible the geometry structure of the manifold [36, 46].

6.5 Experimental results

In this section, we have presented the two dataset's performance of the NR-Isomap method. We have used the S_curve and Swiss roll [45, 85] datasets with noise and compared the NR-Isomap method's results with the Isomap.

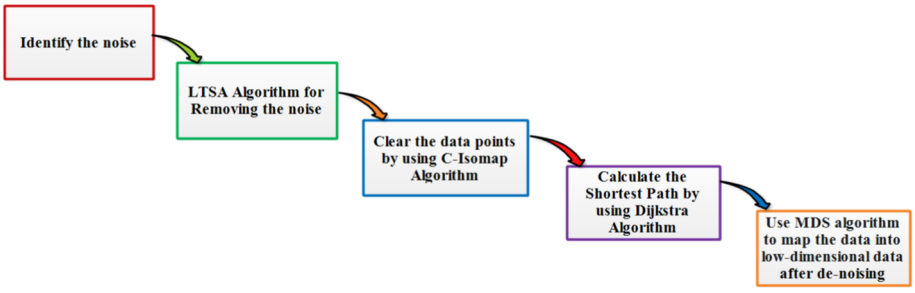

Fig. 24 Main steps of the existing NR-Isomap Method

Algorithm 6 Local tangent space alignment algorithm.

Require: Given Nm-dimensional data points sampled possibly with noise from an underlying d-dimensional data manifold. Let $X_i = \{x_i, \ldots, x_k\}$ be the high-dimensional data space matrix, and the neighborhood parameter is K-neighborhood and ε neighborhood.

Ensure: This algorithm produces N d-dimensional data coordinates $T \in R^{(d \times N)}$ for the manifold constructed from K local nearest neighbours Set neighbourhood parameter, find the neighbourhood N_i of every data point x_i by KNN or ε ball method

1: Extract Local Coordinates
2: **for** (i $\leftarrow$ 1, …, N) **do**
3: Determine K nearest neighbour x_i
4: Compute the d largest eigenvectors $g_1, \ldots, g_d$ of the correlation matrix $X_i - x_i e^T)^T (X_i - x_i e^T)$ and set $G_i = [\frac{e}{\sqrt{K}, g_1, \ldots, g_d}]$
5: **end for**
6: Align Local Coordinates, construct the arranging matrix $\phi = \sum_{i=1}^{N} S_i W_i W \begin{pmatrix} T \\ i \end{pmatrix} S \begin{pmatrix} T \\ i \end{pmatrix}$
7: Calculate the last d+1 singular vector and $T = [u^2, \ldots, u^{(d+1)}]^T$

6.5.1 Result on swiss role dataset with noise values (0.2 to 0.8) and data size (1500)

Figure 25 shows the Swiss roll's original noisy datasets with input data $size$=1500. We have generated the Swiss roll dataset with $noise$ value=0.2 to 0.8, K=40, 60, 80, and 100, and data $size$=1500 in Figs 26 and 27. We have shown the results of Isomap in Fig. 26 (a) and (b) with the value of K=40 and noise=0.2, Fig. 26 (c) and (d) with the value of K=60 and noise=0.4, Fig. 27 (a) and (b) with K=80 and $noise$=0.6, and Fig. 26 (c) and (d) with K=100 and $noise$=0.8. The results of Isomap are noisier when increasing the values of K and $noise$. We have shown the comparative results of NR-Isomap in Figs 26 and 27. In Fig. 26 (a) and (b) use the value of K=40 and $noise$=0.2, Fig. 26 (c) and (d) with a value of K=60 and $noise$=0.4. In Fig. 27 (a) and (b) with (K=80 and $noise$=0.6), and Fig. 27 (c) and (d) with (K=100 and $noise$=0.8), why we have started the value of K=40, the main reason that LTSA is a failure when K<10 because the number of data points is not enough for interaction. When $noise$<0.2, the low-dimensional coordinate can reflect real data points' geometry structure.

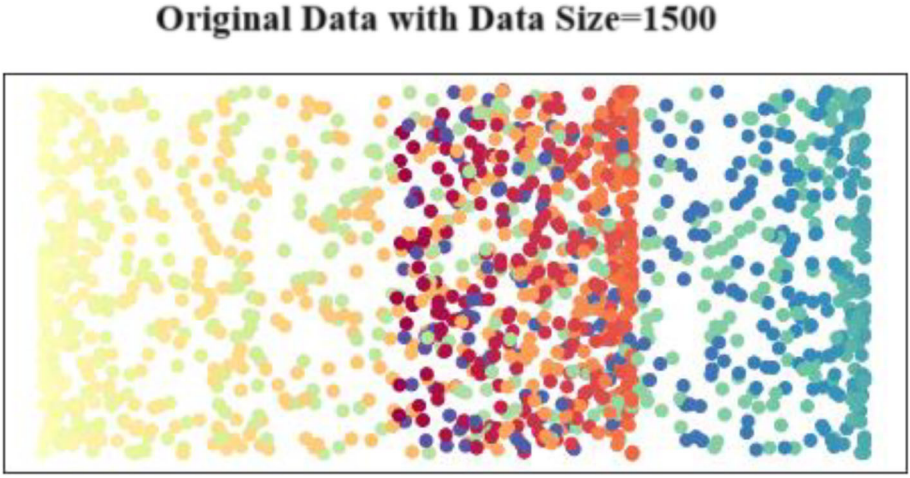

Fig. 25 Noisy swiss roll dataset

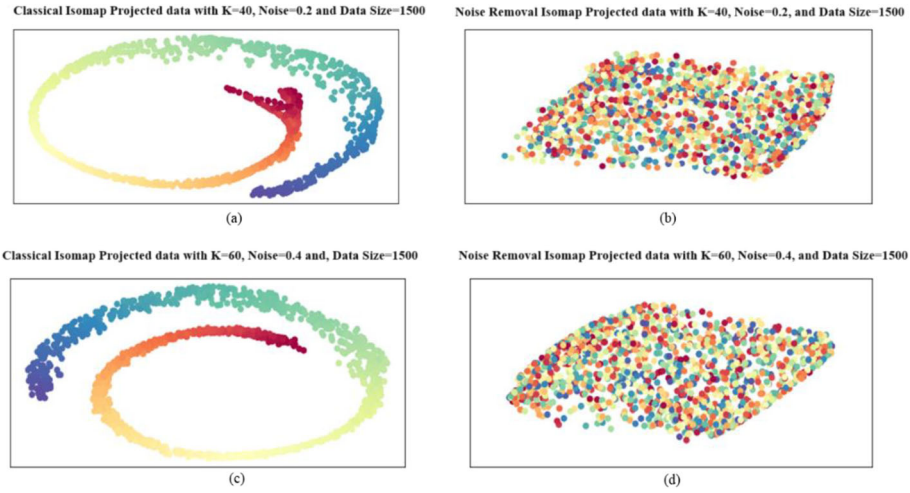

Fig. 26 The comparison between Isomap and NR-Isomap with Swiss Roll Dataset (a) Isomap where; K=40, Noise=0.2, Data size=1500, (b) NR-Isomap where; K=40, Noise=0.2, Data size=1500, (c) Isomap where; K=60, Noise=0.4, Data size=1500, (d) NR-Isomap where; K=60, Noise=0.4, Data size=1500

As the $noise$ value>0.2, the low-dimensional coordinate cannot reflect the geometry's actual data points. The comparison between the Isomap and NR-Isomap methods is clearly shown in the difference in Figs. 26 and 27. The existing NR-Isomap method is much more topologically stable than the Isomap method. The results of our method have effectively reduced the noise from the Swiss roll dataset rather than Isomap. Moreover, it works well when it increases the value of $noise$ and K.

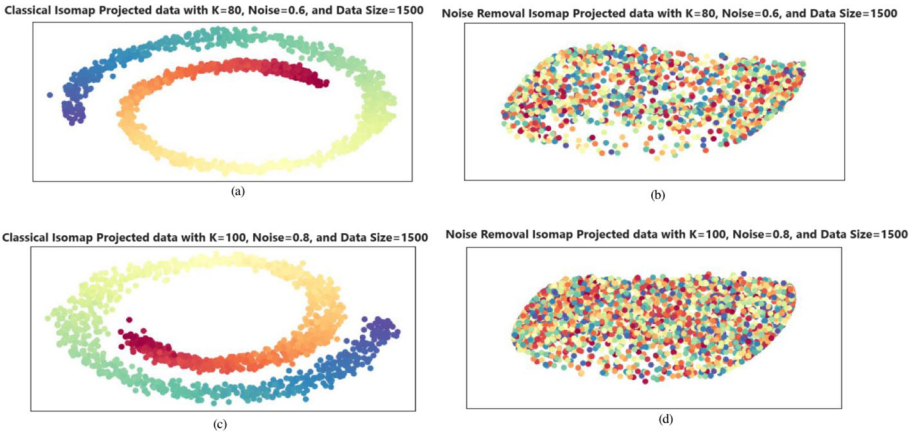

Fig. 27 The Comparison between Isomap and NR-Isomap with Swiss Roll Dataset (a) Isomap where; K=80, Noise=0.6, Data size=1500, (b) NR-Isomap where; K=80, Noise=0.6, Data size=1500, (c) Isomap where; K=100, Noise=0.8, Data size=1500, (d) NR-Isomap where; K=100, Noise=0.8, Data size=1500

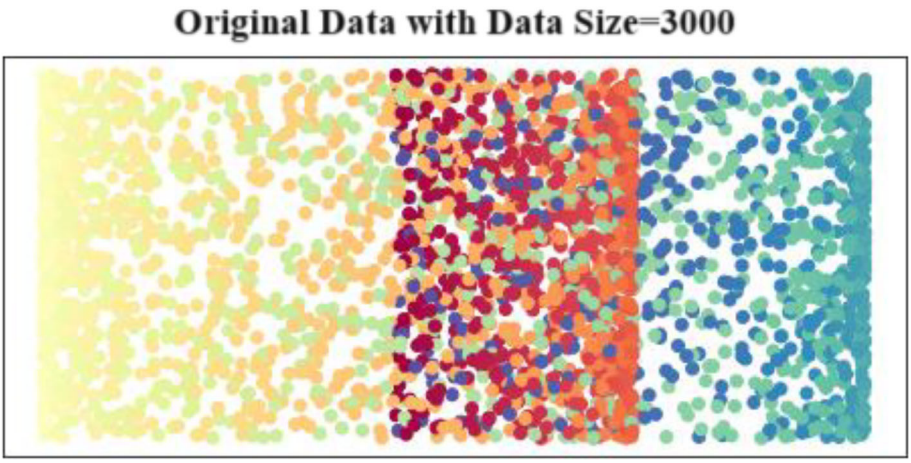

Fig. 28 Noisy swiss roll dataset

6.5.2 Results on swiss roll dataset with noise values (0.2 to 0.8) and data size (3000)

Figure 28 shows the Swiss roll's original noisy datasets with input data $size$ =3000. We have made the Swiss roll dataset with values of $noise$=0.2 to 0.8, K=40, 60, 80, and 100, and data $size$=3000 in Figs 29 and 30. We have shown comparative results of Isomap in Fig. 29 (a) and (b) with use of the value of K=40, $noise$=0.2, Fig. 29 (c) and (d) with the value of K=60, $noise$=0.4, Fig. 30 (a) and (b) use the value of K=80, $noise$=0.6, and Fig. 30 (c) and (d) with K=100, $noise$=0.8. The results of Isomap are noisier while increasing the values of K and $noise$. We have shown comparative results of NR-Isomap in Fig. 29 (a) and (b) with (K=40, $noise$=0.2), Fig. 29 (c) and (d) with (K=60, $noise$=0.4), Fig. 30 (a) and (b) with (K=80, $noise$=0.6), and Fig. 30 (c) and (d) with (K=100, $noise$=0.8). The comparison between Isomap and NR-Isomap shows that our method works in Figs 29 and 30. It is more

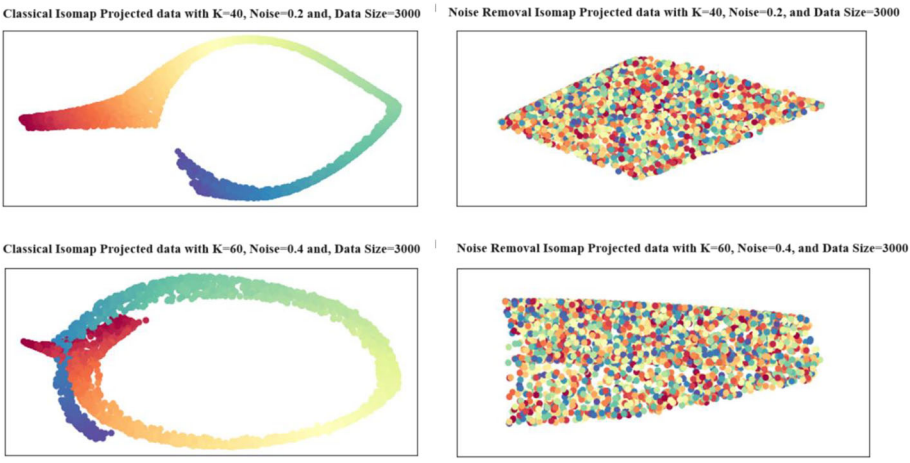

Fig. 29 The comparison between Isomap and NR-Isomap with Swiss Roll Dataset (a) Isomap where; K=40, Noise=0.2, Data size=3000, (b) NR-Isomap where; K=40, Noise=0.2, Data size=3000, (c) Isomap where; K=60, Noise=0.4, Data size=3000, (d) NR-Isomap where; K=60, Noise=0.4, Data size=3000

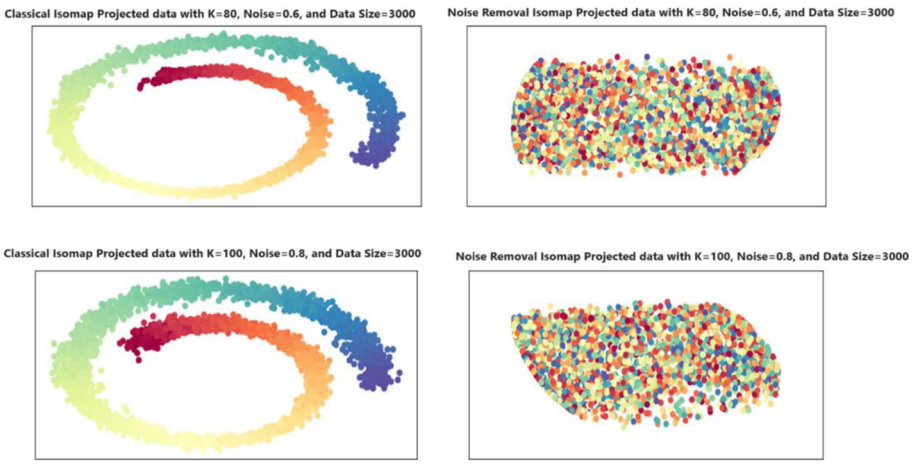

Fig. 30 The comparison between Isomap and NR-Isomap with Swiss Roll Dataset (a) Isomap where; K=80, Noise=0.6, Data size=3000, (b) NR-Isomap where; K=80, Noise=0.6, Data size=3000, (c) Isomap where; K=100, Noise=0.8, Data size=3000, (d) NR-Isomap where; K=100, Noise=0.8, Data size=3000

efficient in stabilizing the topological structure than the Isomap method. The NR-Isomap results have reduced the noise effectively from the Swiss roll dataset rather than the Isomap method. Also, it performs well when it increases the value of $noise$ and K.

6.5.3 Result on S_Curve dataset with noise values (0.2 to 0.8) and data size (1500)

In Fig. 31, we have shown the original datasets of the S_curve with input data $size$ =1500. We have generated the S_curve dataset with values of $noise$=0.2 to 0.8, K=40, 60, 80, and 100, and data $size$=1500 in Fig. 32 to Fig. 33. In Fig. 32 to Fig. 33, we compared NR-Isomap and Isomap methods. We have presented the Isomap results in Fig. 32 (a) and (b) with (K=40 and $noise$=0.2), Fig. 32 (c) and (d) with (K=60 and $noise$=0.4), Fig. 33 (a) and (b) with (K=80 and $noise$=0.6), and Fig. 33 (c) and (d) with (K=100 an $noise$=0.8).

The presented results of Isomap are noisier while increasing the values of $noise$ and K. We have displayed comparative NR-Isomap results in Fig. 32 (a) and (b) with a value of K=40 and $noise$=0.2, Fig. 32 (c) and (d) with (K=60 and $noise$=0.4). In Fig. 33 (a) and (b) with (K=80 and $noise$=0.6), and Fig. 33 (a) and (b) with (K=100 and $noise$=0.8). It is much more topologically stable than the Isomap method. Our method's results have effectually reduced the noise from the S_curve dataset and changed the dataset's shape. Moreover, it performs well when it increases K and $noise$ in the S_curve dataset.

6.5.4 Results on S_Curve dataset with noise values (0.2 to 0.8) and data size (3000)

In Fig. 34, we have presented the original noisy datasets of the S_curve with input data $size$=3000. We have made the S_curve dataset with $noise$ values (0.2 to 0.8), K (40, 60, 80, and 100), and data $size$=3000 in Fig. 35 to Fig. 36. In Fig. 35 to Fig. 36, we compared NR-Isomap and Isomap methods. Fig. 35 (a) and (b) with K=40 and $noise$=0.2, Fig. 35 (c) and (d) with K=60 and $noise$=0.4, Fig. 36 (a) and (b) with K=80 and $noise$=0.6, and Fig. 36

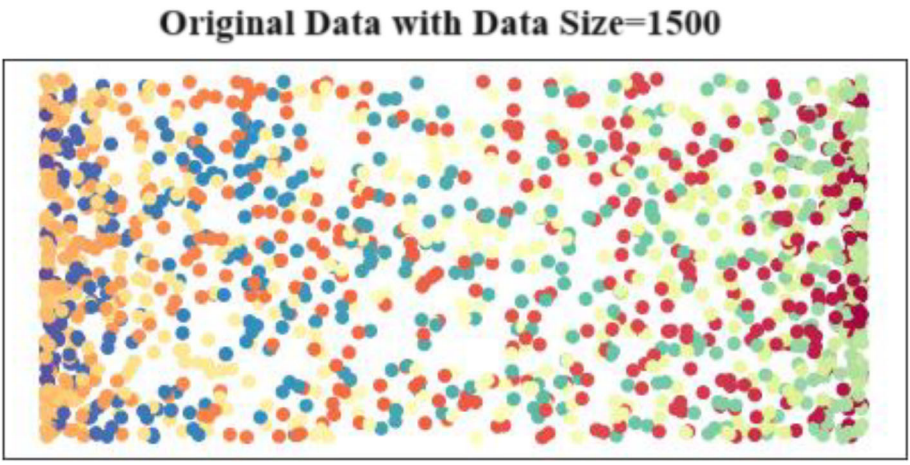

Fig. 31 Noisy S_curve dataset

(c) and (d) with K=100 and $noise$=0.8 are presented with results of Isomap. The displayed results of Isomap are noisier when increasing the values of $noise$ and K, and these results shapes are very messy. Fig. 35 (a) and (b) uses K=40 and $noise$=0.2, Fig. 35 (c) and (d) with K=60 and $noise$=0.4, Fig. 36 (a) and (b) with K=80 and $noise$=0.6, and Fig. 36 (c) and (d) with K=100 and $noise$=0.8 are shown the results of the NR-Isomap method. It is much more stable in the topology structure than the Isomap method. It effectually reduced the noise from the S_curve dataset and changed its shape according to K and $noise$ values. Moreover, it works well by increasing K and noise in the S_curve dataset.

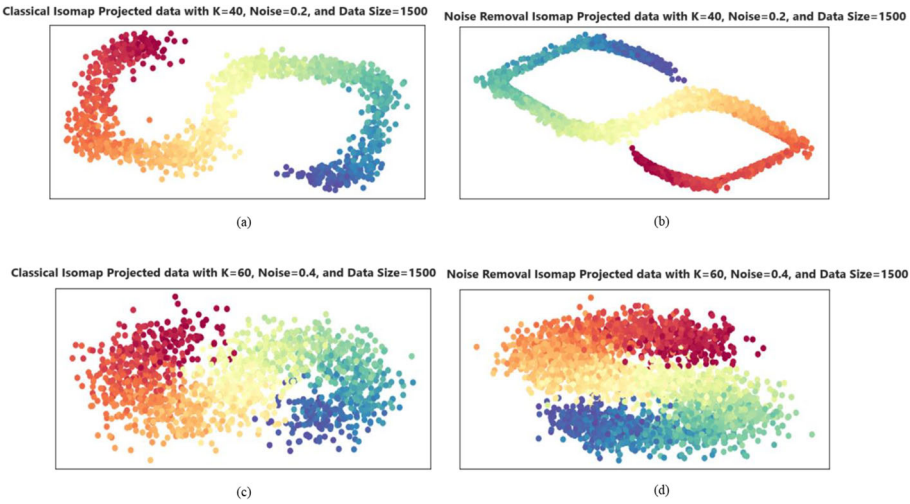

Fig. 32 The comparison between Isomap and NR-Isomap with S_curve Dataset (a) Isomap where; K=40, Noise=0.2, Data size=1500, (b) NR-Isomap where; K=40, Noise=0.2, Data size=1500, (c) Isomap where; K=60, Noise=0.4, Data size=1500, (d) NR-Isomap where; K=60, Noise=0.4, Data size=1500

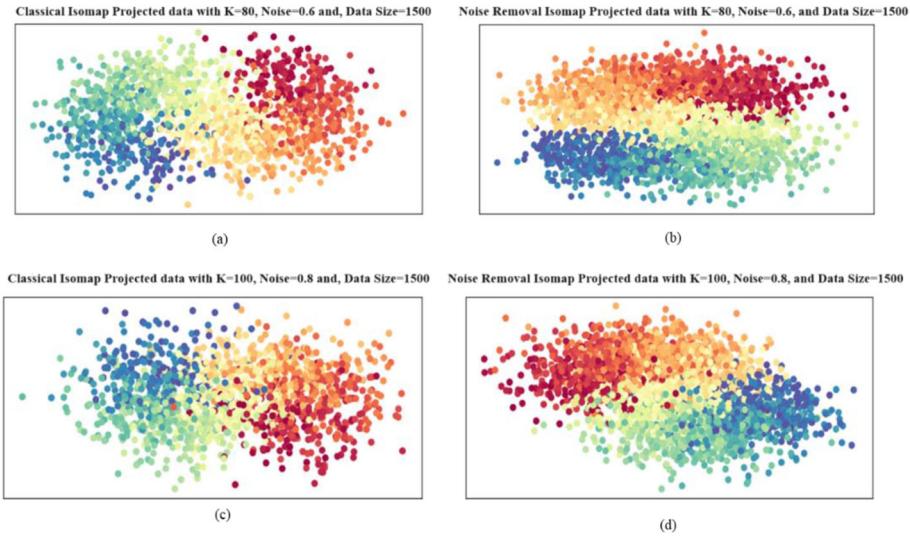

Fig. 33 The comparison between Isomap and NR-Isomap with S_curve Dataset (a) Isomap where; K=80, Noise=0.6, Data size=1500, (b) NR-Isomap where; K=80, Noise=0.6, Data size=1500, (c) Isomap where; K=100, Noise=0.8, Data size=1500, (d) NR-Isomap where; K=100, Noise=0.8, Data size=1500

7 Open problems and future research directions

This section highlights several open problems and potential research directions in manifold learning and the Isomap algorithm. Despite significant progress over the past decade, with applications in pattern recognition, image processing, computer vision, and information retrieval, several unresolved challenges warrant further investigation. These include:

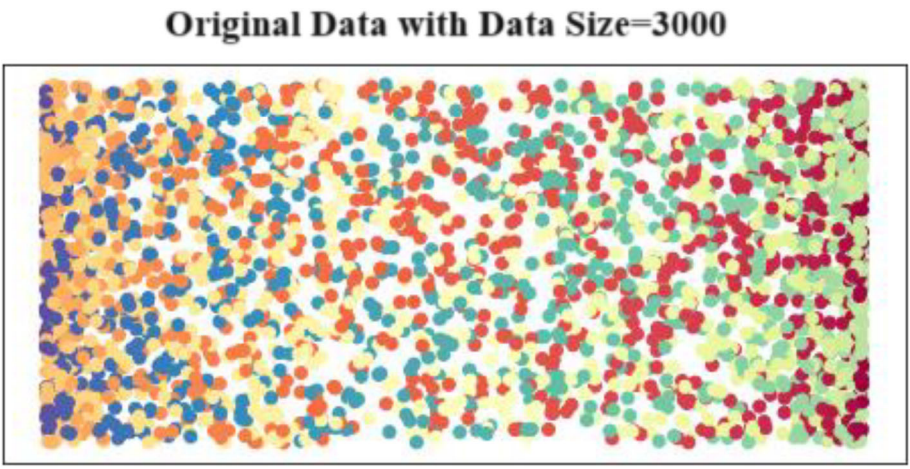

Fig. 34 Noisy S_curve Dataset

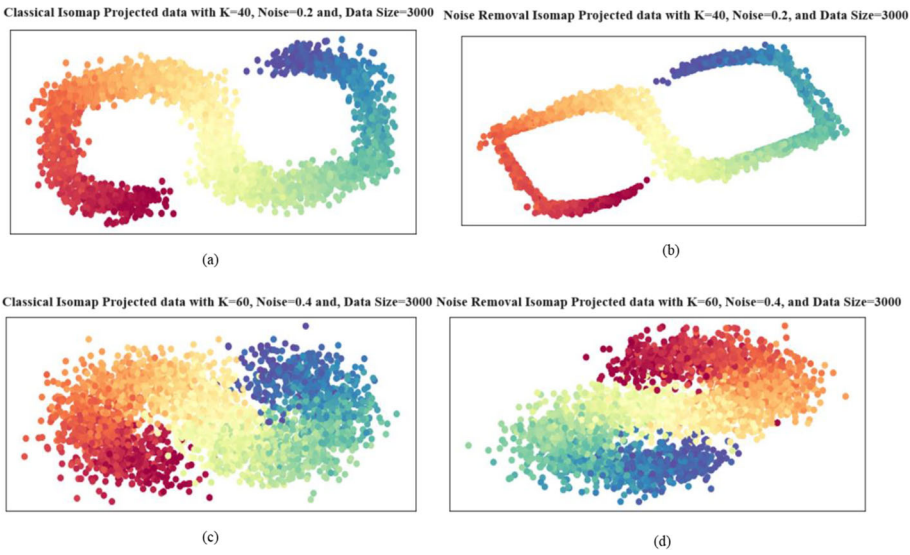

Fig. 35 The comparison between Isomap and NR-Isomap with S_curve Dataset (a) Isomap where; K=40, Noise=0.2, Data size=3000, (b) NR-Isomap where; K=40, Noise=0.2, Data size=3000, (c) Isomap where; K=60, Noise=0.4, Data size=3000, (d) NR-Isomap where; K=60, Noise=0.4, Data size=3000

7.1 Problem 1: In terms of noise manifold learning and Isomap research

The noise detection algorithm presented in this paper is not exclusively tailored for specific manifold learning algorithms. It opens up possibilities for further research in extending its applicability to other manifold learning algorithms. This article primarily focuses on studying noise in manifold learning by considering two common types: uniform and Gaussian noise.

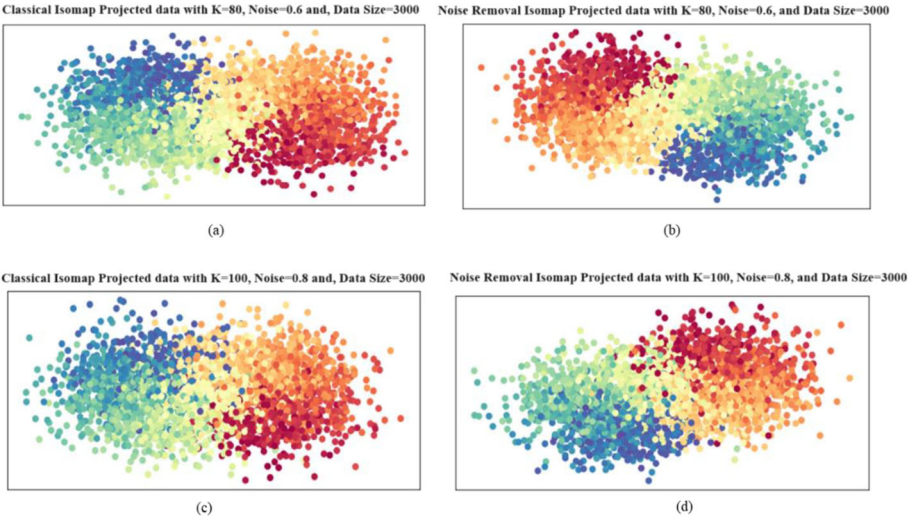

Fig. 36 The comparison between Isomap and NR-Isomap with S_curve Dataset (a) Isomap where; K=80, Noise=0.6, Data size=3000, (b) NR-Isomap where; K=80, Noise=0.6, Data size=3000, (c) Isomap where; K=100, Noise=0.8, Data size=3000, (d) NR-Isomap where; K=100, Noise=0.8, Data size=3000

While these two types of noise are prevalent in real-world scenarios, it is crucial to investigate the performance of different learning algorithms under various noise conditions to understand their robustness comprehensively.

7.2 Problem 2: Parameter selection problem of the Isomap and manifold learning method

The selection of parameters, such as the number of neighbors and Eigendimensions, plays a crucial role in manifold learning algorithms, including the Isomap method. The effectiveness of various learning techniques in identifying low-dimensional manifold structures heavily relies on the appropriate choice of the number of neighbors. However, determining the optimal number of neighbors is challenging due to the influence of data distribution density and spatial structure. Currently, there is a lack of theoretical guidance in this aspect. Another fundamental problem in manifold learning is estimating Eigendimensions, which reflects the topological properties of potential low-dimensional manifolds embedded within high-dimensional observation data. The accuracy of edge dimension estimation significantly impacts the quality of the resulting low-dimensional embedding. While some methods, such as feature mapping and geometric learning, have been proposed, they fail to provide a robust estimate of the edge dimension. Therefore, there is a pressing need for in-depth research on the selection of nearest neighbors and estimating Eigen dimensions in manifold learning algorithms.

7.3 Problem 3: Multi-sub-manifolds learning problem

Most current manifold learning algorithms assume that high-dimensional input data is located in a connected manifold. However, many high-dimensional nonlinear datasets contain multiple sub-manifolds, such as different face data and protein interaction data. The Existing manifold learning methods cannot map all the sub-manifolds included in the dataset to the same low-dimensional space and maintain their respective manifold structure. When applying manifold learning to pattern recognition and image processing, multi-sub-manifold learning problems must be solved. Therefore, it is necessary to study the dimension of high-dimensional data on disconnected manifolds deeply.

7.4 Future research directions

In this section, we present the following as a future research project.

7.4.1 Future work 1: a framework for Isomap algorithm performance evaluation

Algorithm performance evaluation problem involving the mapping results of the manifold learning algorithms. The commonly used Hierarchical method maps high-dimensional data sets into a two-dimensional visualization space and evaluates different datasets' accuracy performance through visual observation. This method can give an objective and fair evaluation. The Approximate Nearest Neighbor (ANN) search has three types: Hierarchical index-based algorithms, graph-based, and Hashing-based algorithms. Hashing-based algorithms are Spectral hashing, Locality Sensitive Hashing (LSH), and Iterative Quantization. ANN searches are used for highly accurate results and provide lower time complexity of the algorithms.

This project will use Hashing-based algorithms for the topological instability problem of the Isomap method.

Locality Sensitive Hashing (LSH): The hashing-based algorithms are a well-known and valuable variant of the LSH function. Various hash functions can be used for the same functions and create a hash value for every data point in LSH [86–89].

7.4.2 Future work 2: manifold learning applied research problem

Most Existing manifold learning algorithms assume that high-dimensional input data resides within a connected manifold. However, numerous nonlinear datasets with high dimensionality consist of multiple sub-manifolds, such as facial or protein interaction data. Conventional manifold learning methods fail to adequately map all the sub-manifolds present in the dataset to a unified low-dimensional space while preserving their manifold structures. Consequently, it becomes crucial to address the challenges of multi-sub-manifold learning when employing manifold learning techniques in pattern recognition and image processing tasks. Therefore, a comprehensive investigation into the dimensionality of high-dimensional data residing on disconnected manifolds becomes imperative.

8 Conclusion

This survey paper comprehensively analyzes the challenges associated with the Isomap method. It encompasses three main aspects: an introduction to the mathematical definition of manifold learning and a comparison with previous Isomap techniques, a detailed examination of various linear and nonlinear methods, and an extensive review of the Isomap method, including its problems and main challenges. We propose three novel existing approaches, FastIsomap, FastIsomapVis, and NR-Isomap, to address the limitations of the Isomap algorithm. The existing methods effectively tackle issues such as incorrect links in the neighborhood graph G, high computational cost, SPD, and the lack of TSP in Isomap. Empirical evaluations on diverse real-world datasets, including Facebook, Twitter, YouTube, SIFT1M, Orkut, Live-Journal, Swiss roll, and S_curve, demonstrate the superior performance of our existing methods in terms of accuracy and computational efficiency compared to the original Isomap algorithm. Notably, our NR-Isomap method exhibits promising results in mitigating noise and short-circuited edges in the TSP of the Isomap. Our efficient existing methods can construct accurate KNN graphs and transform data into low-dimensional space. Moreover, The existing methods can handle large-scale datasets with millions of data points and hundreds of dimensions. Overall, contributions effectively address the challenges posed by the Isomap method, yielding significant performance improvements. However, further research opportunities exist in manifold learning and the Isomap algorithm, which we discuss, open problems, and potential future research directions in this survey paper.

Author Contributions Each author made a significant contribution to the idea and design, to the editing of the document, and to the decision to publish the final edition. In addition, it was agreed that each author would be responsible for all parts of the work and ensure that any concerns about the truthfulness or integrity of any portion of the work would be duly investigated and addressed.

Funding The author(s) received no specific funding grant for this manuscript.

Declarations

Conflicts of interest The authors declare that they have no conflict of interest.

Ethics approval This article contains no studies with human participants or animals performed by authors.

References

1. Sumithra V, Surendran S (2015) A review of various linear and non linear dimensionality reduction techniques. Int J Comput Sci Inf Technol 6(3):2354–2360
2. Van der Maaten L, Hinton G (2008) Visualizing data using t-sne. Journal of machine learning research 9(11)
3. Tang J, Liu J, Zhang M, Mei Q (2016) Visualizing large-scale and high-dimensional data. In: Proceedings of the 25th international conference on world wide web, pp 287–297
4. Lee JA, Verleysen M (2005) Nonlinear dimensionality reduction of data manifolds with essential loops. Neurocomputing 67:29–53
5. Ho HT, Gopalan R (2014) Model-driven domain adaptation on product manifolds for unconstrained face recognition. Int J Comput Vis 109(1):110–125
6. Tenenbaum JB, De Silva V, Langford JC (2000) A global geometric framework for nonlinear dimensionality reduction. Science 290(5500):2319–2323
7. Roweis ST, Saul LK (2000) Nonlinear dimensionality reduction by locally linear embedding. Science 290(5500):2323–2326
8. Belkin M, Niyogi P (2003) Laplacian eigenmaps for dimensionality reduction and data representation. Neural Comput 15(6):1373–1396
9. Gorban AN, Kégl B, Wunsch D.C, Zinovyev A.Y,et al (2007) Principal Manifolds for Data Visualization and Dimension Reduction. vol 58, pp 1–340. Springer Berlin / Heidelberg
10. Tasaki H, Lenz R, Chao J (2019) Dimension estimation and topological manifold learning. In: 2019 International joint conference on neural networks (IJCNN), pp 1–7. IEEE
11. Zheng N, Xue J (2009) Manifold learning. In: Statistical Learning and pattern analysis for image and video processing, pp 87–119. Springer
12. Belkin M, Niyogi P (2001) Laplacian eigenmaps and spectral techniques for embedding and clustering. Nips 14:585–591
13. Torgerson WS (1952) Multidimensional scaling: I. theory and method. Psychometrika 17(4):401–419
14. Fruchterman TMJ, Reingold EM (1991) Graph drawing by force-directed placement. Software - Practice and Experience 21(11):1129–1164
15. Jacomy M, Venturini T, Heymann S, Bastian M (2014) Forceatlas2, a continuous graph layout algorithm for handy network visualization designed for the gephi software. PLOS ONE 9(6):1–18
16. Martin S, Brown WM, Klavans R, Boyack KW (2011) Openord: an open-source toolbox for large graph layout. In: Proceedings of SPIE, the international society for optical engineering, vol 7868, pp 786806
17. Jolliffe I (2011) Principal component analysis,(w:) lovric m. International Encyclopedia of Statistical Science, Springer, New York
18. Lafon S, Lee AB (2006) Diffusion maps and coarse-graining: a unified framework for dimensionality reduction, graph partitioning, and data set parameterization. IEEE Trans Pattern Anal Mach Intell 28(9):1393–1403
19. Nadler B, Lafon S, Coifman RR, Kevrekidis IG (2006) Diffusion maps, spectral clustering and reaction coordinates of dynamical systems.Appl Comput Harmon Anal 21(1):113–127
20. Coifman RR, Lafon S, Lee AB, Maggioni M, Nadler B, Warner F, Zucker SW (2005) Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. Proc Natl Acad Sci U S A 102(21):7426–7431
21. Donoho DL, Grimes C (2003) Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. Proc Natl Acad Sci U S A 100(10):5591–5596
22. Hinton GE, Salakhutdinov RR (2006) Reducing the dimensionality of data with neural networks. Science 313(5786):504–507
23. Mehta S, Zhan B-S, Shen X-J (2019) Weighted neighborhood preserving ensemble embedding. Electronics 8(2):219

24. Bengio Y, Paiement J-f, Vincent P, Delalleau O, Roux N.L, Ouimet M (2003) Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. In: Advances in neural information processing systems 16, vol 16, pp 177–184
25. Choi H, Choi S (2007) Robust kernel isomap. Pattern Recognit 40(3):853–862
26. Balasubramanian M, Schwartz EL (2002) The isomap algorithm and topological stability. Science 295(5552):7–7
27. Li B, Huang D-S, Wang C (2008) Improving the robustness of isomap by de-noising. In: 2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence), pp 266–270
28. Choi H, Choi S (2004) Kernel isomap. Electron Lett 40(25):1612–1613
29. Choi H, Choi S (2005) Kernel isomap on noisy manifold. In: Proceedings. The 4nd international conference on development and learning, 2005., pp 208–213
30. Elhenawy M, Masoud M, Glaser S, Rakotonirainy A (2020) A new approach to improve the topological stability in non-linear dimensionality reduction. IEEE Access 8:33898–33908
31. Yousaf M, Rehman TU, Jing L (2020) An extended isomap approach for nonlinear dimension reduction. SN Comput Sci 1(3):1–10
32. Chang H, Yeung D-Y (2006) Robust locally linear embedding. Pattern Recognit 39(6):1053–1065
33. Mcgill R, Tukey JW, Larsen WA (1978) Variations of box plots. Am Stat 32(1):12–16
34. Feng L, Gao C, Sun T, Wu H (2010) A neighborhood selection algorithm for manifold learning. In: 2010 International conference on computer design and applications, vol 2
35. Chen WW, Mao WJ (2015) An improved isomap algorithm based on lp-centers. In: 2015 International conference on artificial intelligence and industrial engineering, pp 260–263
36. Du C, Zhou S, Sun J, Zhao J (2012) Robust isomap based on neighbor ranking metric. In: International conference on intelligent computing, pp 221–229
37. Kouropteva O, Okun O, Pietikäinen M (2005) Incremental locally linear embedding algorithm. In: SCIA'05 proceedings of the 14th scandinavian conference on image analysis, pp 521–530
38. Kouropteva O, Okun O, Pietikäinen M (2002) Selection of the optimal parameter value for the locally linear embedding algorithm. In: FSKD, pp 359–363
39. Shao C, Huang H (2005) Selection of the optimal parameter value for the isomap algorithm. In: MICAI'05 proceedings of the 4th mexican international conference on advances in artificial intelligence, pp 396–404
40. Saxena A, Gupta A, Mukerjee A (2004) Non-linear dimensionality reduction by locally linear isomaps. In: ICONIP 2003 : international conference on neural information processing, pp 1038–1043
41. Rosman G, Bronstein A, Bronstein M, Kimmel R (2004) Manifold analysis by topologically constrained isometric embedding. Int J Appl Math Comput Sci 1(3):117–123
42. Michels Y, Baudrier E, Mazo L, Tajine M (2019) Density based graph denoising for manifold learning
43. Hong-Yuan W, Xiu-Jie D, Qi-Cai C, Fu-Hua C (2013) An improved isomap for visualization and classification of multiple manifolds. In: ICONIP 2013 proceedings, Part II, of the 20th international conference on neural information processing - Volume 8227, pp 1–12
44. Qu T, Cai Z (2017) An improved isomap method for manifold learning. Int J Intell Comput Cybern 10(1):30–40
45. Qu T, Cai Z (2015) A fast isomap algorithm based on fibonacci heap. In: International conference in swarm intelligence, pp 225–231
46. Lei Y-K, Xu Y, Zhang S-W, Wang S-L, Ding Z-G (2010) Fast isomap based on minimum set coverage. In: ICIC'10 Proceedings of the advanced intelligent computing theories and applications, and 6th international conference on intelligent computing, pp 173–179
47. Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness
48. Yousaf M, Rehman TU, Liao D, Alhusaini N, Jing L (2020) Fastisomapvis: A novel approach for nonlinear manifold learning. IEEE Access 8:199470–199481
49. Jing L, Shao C (2011) Selection of the suitable parameter value for isomap. J Softw 6(6):1034–1041
50. Zhang Z, Chow TW, Zhao M (2012) M-isomap: Orthogonal constrained marginal isomap for nonlinear dimensionality reduction. IEEE Trans Cybern 43(1):180–191
51. Najafi A, Joudaki A, Fatemizadeh E (2016) Nonlinear dimensionality reduction via path-based isometric mapping. IEEE Trans Pattern Anal Mach Intell 38(7):1452–1464
52. Saul LK, Roweis ST (2003) Think globally, fit locally: unsupervised learning of low dimensional manifolds. J Mach Learn Res 4:119–155
53. Zhang Z, Zha H (2005) Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. SIAM J Sci Comput 26(1):313–338
54. Weinberger KQ, Saul LK (2006) Unsupervised learning of image manifolds by semidefinite programming. Int J Comput Vis 70(1):77–90

55. Li M, Yuan B (2005) 2d-lda: A statistical linear discriminant analysis for image matrix. Pattern Recogn Lett 26(5):527–532
56. Raschka S (2014) Linear discriminant analysis bit by bit. Disponible en: https://sebastianraschka.com/Articles/2014_python_lda.html
57. Dzidolikaitė A (2015) Genetic algorithms for multidimensional scaling. Mokslas-Lietuvos ateitis/Science-Future of Lithuania 7(3):275–279
58. Žilinskas A, Žilinskas J (2006) On multidimensional scaling with euclidean and city block metrics. Technol Econ Dev Econ 12(1):69–75
59. Hougardy S (2010) The floyd-warshall algorithm on graphs with negative cycles. Inf Process Lett 110(8–9):279–281
60. Jo J, Seo J, Fekete J-D (2017) A progressive kd tree for approximate k-nearest neighbors. In: 2017 IEEE workshop on data systems for interactive analysis (DSIA), pp 1–5. IEEE
61. Silpa-Anan C, Hartley R (2008) Optimised kd-trees for fast image descriptor matching. In: 2008 IEEE conference on computer vision and pattern recognition, pp 1–8. IEEE
62. Dong W, Moses C, Li K (2011) Efficient k-nearest neighbor graph construction for generic similarity measures. In: Proceedings of the 20th international conference on world wide web, pp 577–586
63. Toolbox GO (2016) User's guide. Matlab R2016a, The MathWorks. Inc
64. Gulraj M, Ahmad N (2016) Mood detection of psychological and mentally disturbed patients using machine learning techniques. Int J Comput Sci Netw Secur (IJCSNS) 16(8):63
65. Leskovec J, Krevl A (2014) SNAP Datasets: Stanford Large Network Dataset Collection. MI, USA, Ann Arbor
66. Amsaleg L, Jegou H (2010) Datasets for approximate nearest neighbor search
67. Ho TK (1998) Nearest neighbors in random subspaces. In: Joint IAPR international workshops on statistical techniques in pattern recognition (SPR) and structural and syntactic pattern recognition (SSPR), pp 640–648. Springer
68. Lowe DG (1995) Similarity metric learning for a variable-kernel classifier. Neural Comput 7(1):72–85
69. Vapnik VN (1995) The nature of statistical learning. Theory
70. Vapnik VN, Vapnik V (1998) Statistical Learning Theory, vol 1. Hoboken. NJ, USA: Wiley
71. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. Nature 323(6088):533–536
72. Witten IH, Frank E, Hall MA, Pal C, DATA M (2005) Practical machine learning tools and techniques. In: DATA MINING, vol 2, pp 4
73. Geng X, Zhan D-C, Zhou Z-H (2005) Supervised nonlinear dimensionality reduction for visualization and classification. IEEE Trans Syst, Man, and Cybern, Part B (Cybernetics) 35(6):1098–1107
74. Sivakumar S, Chandrasekar C (2014) Modified dijkstra's shortest path algorithm. Int J Innov Res Comput Commun Eng 2(11):6450–6456
75. Zhan FB (1997) Three fastest shortest path algorithms on real road networks: Data structures and procedures. J Geogr Inf Decis Anal 1(1):69–82
76. Abujassar R, Ghanbari M (2011) Efficient algorithms to enhance recovery schema in link state protocols. arXiv:1108.1426
77. Eneh A, Arinze U (2017) Comparative analysis and implementation of dijkstra's shortest path algorithm for emergency response and logistic planning. Niger J Technol 36(3):876–888
78. Xiao-Yan L, Yan-Li C (2010) Application of dijkstra algorithm in logistics distribution lines. In: Third international symposium on computer science and computational technology (ISCSCT'10), Jiaozuo, PR China, pp 048–050. Citeseer
79. Wang H, Yu Y, Yuan Q (2011) Application of dijkstra algorithm in robot path-planning. In: 2011 Second international conference on mechanic automation and control engineering, pp 1067–1069. IEEE
80. Rennie J, Lang K (2008) The 20 newsgroups data set. Available in web page¡ URL: http://qwone.com/jason/20Newsgroups
81. Dua D (2019) Graff C (2019) UCI Machine Learning Repository, University of California, School of Information and Computer Science. Irvine, CA
82. Karlik B, Al-Bastaki Y (2004) Real time monitoring odor sensing system using omx-gr sensor and neural network. WSEAS Trans Electron 1(2):337–342
83. Wang J (2012) Local tangent space alignment. Geometric structure of high-dimensional data and dimensionality reduction, pp 221–234
84. Orts Gómez FJ, Ortega López G, Filatovas E, Kurasova O, Garzón GEM (2019) Hyperspectral image classification using isomap with smacof. Informatica 30(2):349–365
85. Tenenbaum JB et al (1998) Mapping a manifold of perceptual observations. Adv Neural Inf Process Syst 10:682–688

86. Indyk P, Motwani R (1998) Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proceedings of the thirtieth annual ACM symposium on theory of computing, pp 604–613
87. Gionis A, Indyk P, Motwani R et al (1999) Similarity search in high dimensions via hashing. Vldb 99:518–529
88. Andoni A, Indyk P (2006) Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In: 2006 47th Annual IEEE symposium on foundations of computer science (FOCS'06), pp 459–468. IEEE
89. Hyvönen V, Pitkänen T, Tasoulis S, Jääsaari E, Tuomainen R, Wang L, Corander J, Roos T (2015) Fast k-nn search. arXiv:1509.06957