Discover Artificial Intelligence

Review

Clustering graph data: the roadmap to spectral techniques

Rahul Mondal[1,6] · **Evelina Ignatova**[1] · **Daniel Walke**[1,4] · **David Broneske**[2] · **Gunter Saake**[1] · **Robert Heyer**[3,5]

Received: 12 June 2023 / Accepted: 2 January 2024
Published online: 22 January 2024
© The Author(s) 2024 OPEN

Abstract
Graph data models enable efficient storage, visualization, and analysis of highly interlinked data, by providing the benefits of horizontal scalability and high query performance. Clustering techniques, such as K-means, hierarchical clustering, are highly beneficial tools in data mining and machine learning to find meaningful similarities and differences between data points. Recent developments in graph data models, as well as clustering algorithms for graph data, have shown promising results in image segmentation, gene data analysis, etc. This has been primarily achieved through research and development of algorithms in the field of spectral theory, leading to the conception of spectral clustering algorithms. Spectral clustering algorithms have been one of the most effective in grouping similar data points in graph data models. In this paper, we have compiled 16 spectral clustering algorithms and compared their computational complexities, after an overview of graph data models and graph database models. Furthermore, we provided a broad taxonomy to classify most existing clustering algorithms and discussed the taxonomy in detail.

Keywords Graph data · Clustering · Laplacian · Spectral

1 Introduction

Graph data models are useful to store, process and analyse highly interlinked data [1]. This is achieved through the use of graph theory to store data in the form of nodes and edges [2]. With the recent rise in the popularity of graph databases, which rely on graph data models to store and query data, there is a growing need to incorporate state-of-the-art learning algorithms to analyse data in graph data models. This incorporation enables the extraction of meaningful information that might otherwise be hidden when analysed in a tabular structure. Employing unsupervised and supervised learning techniques is highly effective for analysing data across several domains, e.g. to study social networks [3], physical systems [4], proteomics knowledge graphs [5], etc [6].

One of the most commonly used unsupervised learning algorithms is clustering, which is widely used by data analysts and domain experts to group similar instances and explore hidden structures in a wide spectrum of fields, ranging from engineering, computer science and medical sciences to social sciences and economics as well [2]. The challenges and

✉ Evelina Ignatova, evelina.ignatova@st.ovgu.de; Rahul Mondal, rahul.mondal@thi.de; Daniel Walke, daniel.walke@ovgu.de; David Broneske, broneske@dzhw.eu; Gunter Saake, saake@ovgu.de; Robert Heyer, robert.heyer@uni-bielefeld.de; robert.heyer@isas.de | [1]Faculty of Computer Science, Otto-von-Guericke-University, Universitätsplatz 2, 39106 Magdeburg, Saxony-Anhalt, Germany. [2]German Center for Higher Education Research and Science Studies (DZHW), Lange Laube 12, 30159 Hannover, Lower Saxony, Germany. [3]Faculty of Technology, Bielefeld University, Universitätsstraße 25, 33615 Bielefeld, North Rhine-Westphalia, Germany. [4]Faculty of Process and Systems Engineering, Otto-von-Guericke-University, Universitätsplatz 2, 39106 Magdeburg, Saxony-Anhalt, Germany. [5]Multidimensional Omics Analyses Group, Leibniz-Institut für Analytische Wissenschaften – ISAS – e.V., Bunsen-Kirchhoff-Straße 11, 44139 Dortmund, North Rhine-Westphalia, Germany. [6]Faculty of Computer Science, Technische Hochschule Ingolstadt, Esplanade 10, 85049 Ingolstadt, Bavaria, Germany.

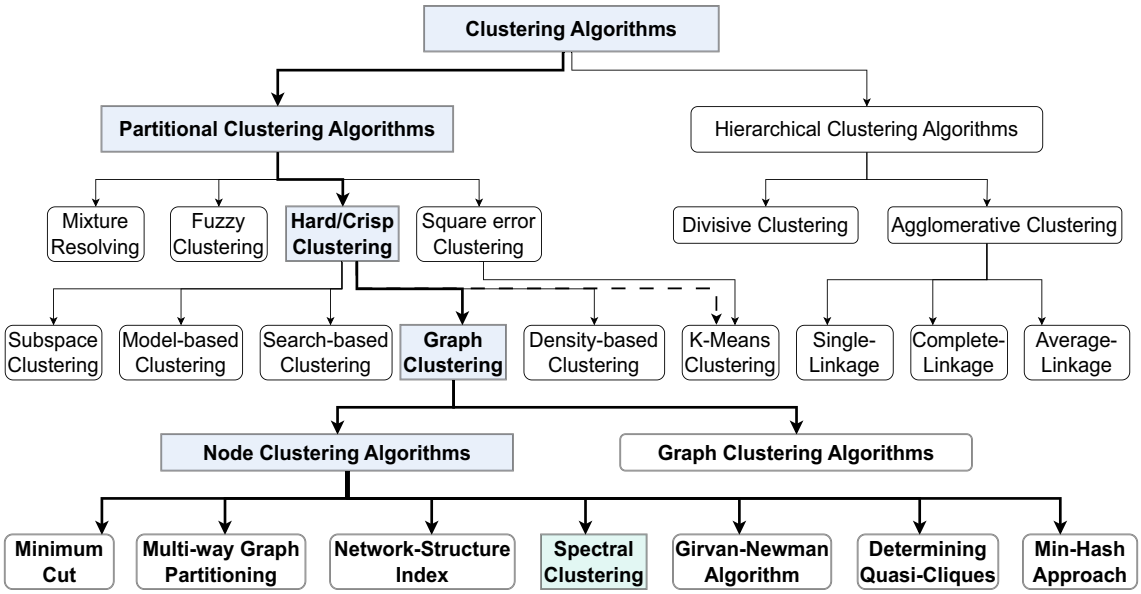

Fig. 1 Taxonomy of clustering algorithms [7, 8]. Coloured nodes and dark edges highlight the roadmap leading to spectral clustering. The dotted arrow denotes that k-means is also a hard/crisp clustering, in addition to being a squared error reduction-based clustering algorithm as well

opportunities of graph data have led to the development of specialized clustering algorithms designed specifically for graph data. These algorithms include spectral clustering which often outperforms basic clustering algorithms such as K-means, hierarchical, etc. [9].

The popularity of graph data is on the rise with the development of graph database management systems, e.g. AllegroGraph, ArangoDB, InfiniteGraph, Neo4J [10]. The flexibility of the graph data structures allows novel possibilities for data exploration, and consequently, knowledge discovery. As a result, clustering algorithms designed for graph data models, e.g. spectral clustering algorithms, are gaining momentum along with the applications of graph data and databases.

A popular class of clustering algorithms, designed specifically for graph data models, is known as spectral clustering. Spectral clustering utilizes eigendecomposition to represent and group data into clusters [11], and its conceptualization dates back to 1973 [12]. This survey analyses popular spectral clustering algorithms, which have been widely discussed in the scientific community due to their high efficiency in applications such as image segmentation, analysing patterns in gene expression data or proteomics data. We initially provide a roadmap (see Fig. 1) to navigate through the clustering paradigm till spectral clustering is reached, upon which we elaborate on 16 primary spectral clustering algorithms and conclude with a comparison of their complexities and applications. Since several variations of these algorithms were developed, we will concentrate on discussing most of them extensively in a single survey.

In this paper, we have provided a comprehensive overview of the following topics:

- Background: Graph theory, database models, and cluster analysis (Sect. 3)
- Types of clustering algorithms (Sect. 3.3)
- Clustering graph data: Graph and node clustering (Sect. 4.1 and 4.2)
- Spectral clustering algorithms (Sect. 4.3)

1.1 Related work

There are several clustering techniques and comprehensive analyses of clustering algorithms for e.g., clustering algorithms in general by Ezugwu et al. [7], clustering algorithms for graph data by Aggarwal et al. [8], spectral clustering algorithms by Nascimento et al. [11] and Verma et al. [13]. The conception of new spectral clustering algorithms is more suited for specific tasks rather than generic data. Karim et al. [14] have demonstrated in their survey on deep learning-based clustering approaches their usefulness in the field of bioinformatics. Another similar work by Qi et al. [15] has surveyed various clustering and classification methods specifically for single-cell RNA-sequencing data. Several recent

developments in the domain of spectral clustering require an elaborate survey of significant techniques and a roadmap to trace the development of the use of eigenvectors for clustering.

2 Definitions and notations

Graph: A graph is represented by, G = (V,E), where V denotes a set of nodes (vertices) and E denotes a set of edges (relationships) between the nodes. Edges can be weighted or unweighted. Edges should be undirected for spectral clustering algorithms (von Luxburg [9]). There are three different methods to transform data points into a similarity graph, for spectral clustering [9] as shown in Fig. 2:

- Fully connected graph [16, 17]: Any data points with positive adjacency values can be connected to form a graph which is only useful when the similarity function itself can model local neighbourhoods, e.g.:

$$A_{ij} = exp(-||s_i - s_j||^2/2\sigma^2) \quad (1)$$

  where A_{ij} is the affinity between s_i and s_j; $s_i - s_j$ is the distance between s_i and s_j; σ is the scaling parameter
- ϵ-neighbourhood graph [18]: Data points are connected based on a threshold, ϵ. Edges with weights lower than the ϵ are discarded to form a ϵ-neighbourhood graph from a fully-connected graph.
- k-neighbourhood graph [19]: Formed using the k-nearest neighbour algorithm resulting in either k-nearest neighbour or mutual k-nearest neighbour graph, depending on how the vertices were connected. k denotes the minimum number of points required to define a local neighbourhood.

Proximity measure: Measure of distance, similarity, dissimilarity and/or adjacency between vertices/nodes/data points derived from their attributes. Clustering algorithms always require calculating proximity measures, among the given data points, as their primary step. Popular examples (Mehta et al. [20]) in the context of clustering can be broadly categorized into two types: metric and non-metric (see Table 1). Metric proximity measures satisfy properties such as non-negativity, symmetry, and triangle inequality. On the other hand, non-metric proximity measures may violate symmetry and/or triangle inequality.

Distance matrix: Square matrix (refer Fig. 3a) representing the distance between data points based on a distance measure, e.g. Euclidean, Manhattan, etc. The diagonal values are 0 which signifies the lowest possible distance value.

Similarity matrix: Square matrix (refer Fig. 3b) representing the similarity between data points based on a similarity measure, e.g. Dice, Cosine, etc. Diagonal value 1 represents the highest possible similarity value. However, the diagonal of a square matrix containing dissimilarity between data points may contain 0 as the lowest possible dissimilarity value.

Adjacency (affinity) matrix: Square matrix (refer Fig. 3c) representing the adjacency (also affinity or node similarity) between points/nodes in a graph, G (Hogben [29]). It can be of two types—unweighted and weighted.

For weighted adjacency matrix: A = $[\alpha_{i_j}]$, where, α_{i_j} is an edge of graph, G

For unweighted adjacency matrix: A = $[\alpha_{i_j}]$, where, $\alpha_{i_j} = 1$ if {i,j} is an edge of graph, G and $\alpha_{i_j} = 0$ otherwise.

Diagonal (degree) matrix: Square matrix containing the degree of each node in its diagonal (Hogben [29]) which represents the number of edges, connected to each node in the graph.

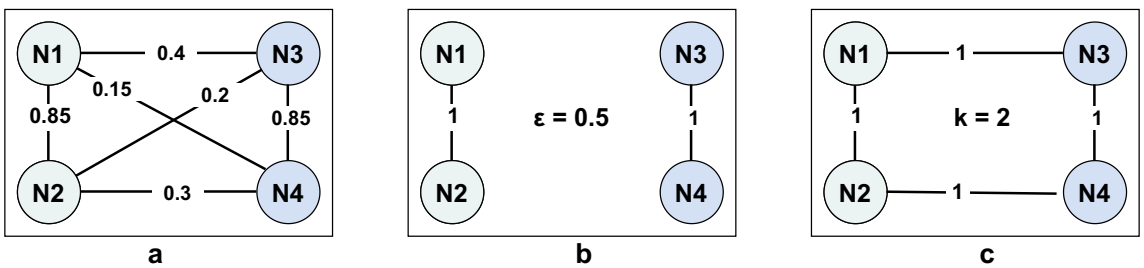

Fig. 2 Types of neighbourhood graphs. **a** Fully-connected graph; **b** ϵ-neighbourhood graph; **c** k-neighbourhood graph for weighted graphs, all the edge weights in **b** and b would not be replaced by 1

Table 1 A generic overview of popular proximity measures used in clustering

Proximity	Type	Equation
Euclidean distance [21]	Metric	$\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$
Manhattan distance [21]	Metric	$\sum_{i=0}^{n} \lvert x_i - y_i \rvert$
Minkowski distance [22]	Non-metric	$\left(\sum_{i=1}^{n} \lvert x_i - y_i \rvert^p\right)^{\frac{1}{p}}$
Chebyshev distance [23]	Metric	$max_{i=1,2,\ldots n} \lvert x_i - y_i \rvert$
Mahalanobis distance [22]	Metric	$\sqrt{(x_i - y_i)^T V^{-1} (x_i - y_i)}$
Hamming distance [22]	Non-metric	$\sum_{i=0}^{n} \lvert x_i - y_i \rvert$
Canberra distance [23]	Metric	$\sum_{i=1}^{n} \frac{\lvert x_i - y_i \rvert}{\lvert x_i \rvert + \lvert y_i \rvert}$
Pearson correlation [24]	Non-metric	$E(xy)/\sigma_x \sigma_y$
Jaccard coefficient [25]	Metric	$\frac{\lvert N(x) \cap N(y) \rvert}{\lvert N(x) \cup N(y) \rvert}$
Dice coefficient [26]	Non-metric	$\frac{2\lvert A \cap B \rvert}{\lvert A \rvert + \lvert B \rvert}$
Cosine similarity [27]	Non-metric	$\frac{\vec{x}.\vec{y}}{\lvert \vec{x} \rvert + \lvert \vec{y} \rvert}$
Bray–Curtis dissimilarity [28]	Non-metric	$1 - (2 * C_{ij})/(S_i + S_j)$
Kullback–Leibler divergence [22]	Non-metric	$\sum_{i=1}^{n} x_i \times log_2\left(\frac{x_i}{y_i}\right)$

n = data set size, x_i, y_i = vectors with *i* elements, p = scaling factor, V = covariance matrix, E = cross-correlation, σ = standard deviation, N(x)/N(y) = set of vertices that form the "neighbourhood" of a single vertex x/y, ∩ = set intersections, ∪ = set union, A = ground-truth, B = predicted label, C_{ij} = sum of lesser values for species found in sites i and j, S_i = sum of species found at site i, S_j = sum of species found at site j

distance	N1	N2	N3
N1	0	d1	d2
N2	d1	0	d3
N3	d2	d3	0

a

similarity/ dissimilarity	N1	N2	N3
N1	1 / 0	s1	s2
N2	s1	1 / 0	s3
N3	s2	s3	1 / 0

b

adjacency	N1	N2	N3
N1	0	a1	a2
N2	a1	0	a3
N3	a2	a3	0

c

Fig. 3 Matrix representation of **a** distance, **b** similarity (affinity) and **c** adjacency matrices for nodes N1, N2 and N3

D = diag(deg_G1,...,deg_Gn), where, n = number of nodes/data points and, deg = degree of a node (number of edges connected to a node).

Laplacian matrix: Spectral clustering algorithms require adjacency and degree matrix to create the Laplacian matrix of the input graph which acts as a primary input to most. Commonly used types in spectral clustering [9, 30] are:

- Unnormalized, L = D – A
- Normalized:
  - Symmetric, $L_{sym} = D^{-1/2} L D^{-1/2}$
  - Random Walk, $L_{rw} = D^{-1} L$
- Relaxed, $L_\rho = L - {}_\rho D$
  where D = Diagonal and A = Adjacency matrix, and ρ = relaxation parameter.

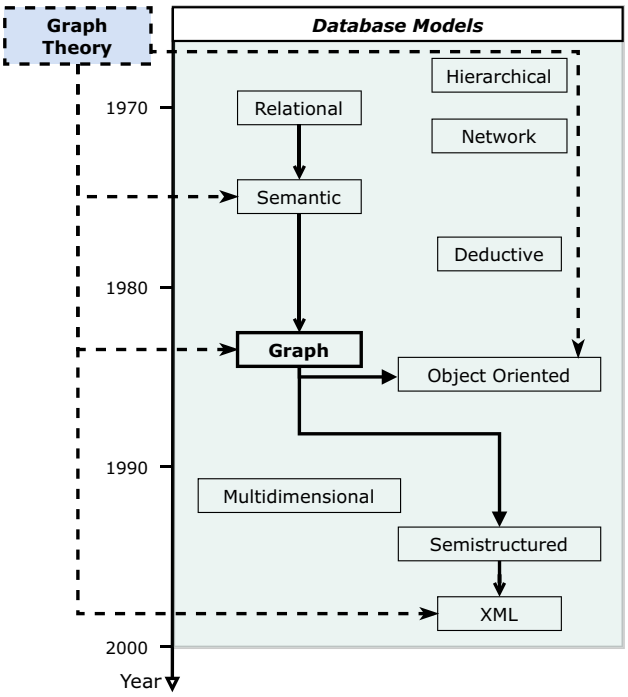

Fig. 4 Evolution of database models. Arrows denote influence—dotted arrows represent the influence of graph theory on various database models [32]

Table 2 Graph vs relational database models [36, 37]

	Relational	Graph
Transaction model	ACID	BASE (Neo4j complies ACID as well)
Query language	SQL	Cypher, Gremlin, SPARQL, GQL
Scalability	Vertical	Horizontal
Integrity constraints	Yes	Yes
Flexibility	Less mutable schema	Easily mutable schema
Support	High	Inferior
Ease of programming	Easy	Difficult
Security	Extensive for ACL-based	Only at application level

3 Background

3.1 Graph theory

Graph theory is a branch of discrete mathematics that deals with the study of mathematical structures to model entities and relationships between them, in the form of nodes/vertices and relations/edges, respectively. It has been discussed by Pal Singh et al. [31], along with its wide spectrum of applications in database design, software engineering, circuit designing, network designing, and visual interfaces. Graph theory influenced the conception of several database models, such as semantic, object-oriented, graph, and XML, as shown in Fig. 4. Some popular data structures influenced by graph theory are trees, linked lists, etc. that can be used to model graphs.

3.2 Graph database models

Graph databases primarily provide storage and querying of data stored in graph data models. Additionally, available plug-ins [32] can provide features such as conceptual visualization, e.g. Neo4j bloom [33], data analytics, e.g. Graph Data Science library [34], Decision Tree Plug-in [35]. Hence the full potential of graph data models could be realized on a Graph Database Management System such as Neo4j [36], AllegroGraph [10]. Chad et al. [36] have evaluated Neo4j,

a Graph Database Management System, against Relational Database Management System as shown in Table 2. The flexibility of data schema in a graph database represents the added benefit over a relational database. Regarding scalability, it could be argued that relational databases perform better regarding data distribution across several machines, as discussed by Pokorny [1]. However, scalability on large datasets is not an issue for graph databases [1].

3.3 Cluster analysis

Clustering (cluster analysis), is a process of grouping data into distinct classes so that objects with similar attributes and/or characteristics are grouped in the same class/clusters [7]. It is classified as an unsupervised learning algorithm where the goal is to find meaningful patterns from underlying unlabeled data [38].

Traditional clustering algorithms such as K-means, DBSCAN, and agglomerative clustering, suffer when dealing with high-dimensional data since most often Euclidean distance alone is used as the distance/proximity measure between data points which fails at accurately portraying the relative positions of data points at high dimensions [39]. Spectral clustering algorithms overcome this (graphs are non-Euclidean data structures [40]) through the calculation of eigenvalues and eigenvectors from Euclidean distances of the graph Laplacian matrix to partition the graph in the eigenspace [8].

Clustering algorithms could be broadly generalized into two categories: partitional and hierarchical clustering (Celebi et al. [41]). The former partitions data points according to a pre-defined number of groups, while the latter hierarchically assigns data points as groups of subgroups, until all points belong to one cluster (bottom-up) or individual clusters (top-down). A brief comparison between partitional and hierarchical clustering algorithms is provided in Table 3. While hierarchical clustering is generously illustrative to have an elaborate overview of the cluster formation, which acts as a huge advantage to realize the similarity between data points in sub-clusters, it comes at the cost of higher time and space complexity (Garima et al. [42]) than partitional clustering.

Sum of squares of error (SSE) minimization: The most commonly used partitional clustering technique, K-means, optimizes SSE of clusters, while Ezugwu et al. [7] also labels it as a hard clustering technique. It has the advantage of being easily implemented on large datasets at a considerably low run time. The results are easily interpretable, which benefits the user in having a general overview of the data and possible clusters.

Fuzzy: Fuzzy clustering involves assigning the degree of membership, for each data point to more than one cluster, e.g. fuzzy c-means algorithm [48].

Mixture resolving: Mixture resolving methods, according to Grira et al. [49], assume that data points belong to one of several distributions. Expectation maximization (EM) is an iterative approach that aims to find the maximum likelihood estimates of the parameters [50] and is used in this case for parameter estimation.

Hard clustering: Hard clustering, e.g. K-means, groups data into prespecified k non-overlapping groups, without a hierarchy [7].

- Density-based clustering algorithms such as DBSCAN (Ester et al. [51]), OPTICS (Ankerst et al. [44]), DENCLUE (Hinneburg et al. [52]) provide better performance than K-means by handling arbitrary shapes and detecting outliers.
- Subspace clustering algorithms can be categorized into top-down and bottom-up algorithms. PART (Cao et al. [53]) and PROCLUS (Aggarwal et al. [54]) are top-down algorithms, in which the whole set of dimensions is used to find an initial grouping, and the subspaces of each cluster are assessed. On the other hand CLIQUE (Agrawal et al. [55]), and MAFIA (Nagesh et al. [56]) are bottom-up subspace algorithms, in which first dense areas in low-dimensional spaces are identified, then by combining them, clusters are created (Gan et al. [57]).
- In model-based clustering such as COOLCAT (Barbará et al. [58]) and STUCCO (Bay et al. [59]), it is presupposed that the data are produced by a combination of probability distributions, each of which components represents a distinct cluster (Gan et al. [57]).
- Search-based algorithms such as Genetic Algorithms (Holland et al. [60]), Al-Sultan's Method (Barbara et al. [58]) work towards globally optimal clustering to fit the data. Compare to, for example, fuzzy clustering, search-based algorithms do not stop at a local optimum partition (Gan et al. [57]).
- Other algorithms inside the class of hard clustering are termed Miscellaneous Algorithms. Examples include (Gan et al. [57]) Time Series Clustering Algorithms in which data is usually classified into two categories—much individual time series and a single time series; Streaming Algorithms—tremendous amounts of data, including network data, temperature data, and satellite imagery data; Transaction Data Clustering Algorithms—for transaction data (market basket data) etc.

Table 3 Comparison of commonly used clustering algorithms [41, 42]

Type of clustering	Parameters	Complexity (time)	Pros	Cons
Partitional				
Hard				
SSE minimization (K-means)	Number of clusters	$O(n^2)$ [43]	Can efficiently handle large amount of data	Cannot handle non-circular clusters
Density-based (OPTICS)	Minimum points to form a cluster	$O(n^3)$ [44]	Can handle arbitrary shapes of data	Computationally expensive
Graph (spectral)	Number of clusters, neigbourhood graph	$O(n^3)$ [45]		
Fuzzy (fuzzy c-means)	Fuzzifier (m) membership value (u)	O(NCT) [46]	Data points can belong to multiple clusters	Performance depends on initialization
Hierarchical				
Agglomerative (single-linkage)	Linkage criterion	$O(n^3)$ [47]	Do not require initialization	Computationally expensive

n = number of nodes, N = number of links, C = Number of link clusters, T = Number of iterations

- Graph clustering is another type of hard clustering, tailored to cluster data stored in graph data structures (Nascimento et al. [11]). Algorithms of graph clustering can be broadly classified into two categories—graph and node clustering algorithms.

4 Clustering graph data: graph and node clustering algorithms

4.1 Graph clustering algorithms

Graph clustering algorithms are concerned with clustering several graphs rather than one, each with a set of nodes and edges, based on their underlying structure, and could be discussed either in the context of graph data as well as semi-structured data, e.g. XML data. Some popular approaches in this regard are:

- Structural distance-based approach, e.g. XClust (Lee et al. [61]).
- Structural summary-based approach (Dalamagas et al. [62]).
- The XProj approach (Aggarwal et al. [63]).

The use of eigenvectors to represent and cluster data points or graph nodes was made popular by the conception of spectral clustering, which is a form of a node clustering algorithm.

4.2 Node clustering algorithms

Node clustering algorithms use a distance function to measure proximity between data points or nodes, of a multi-dimensional dataset (Aggarwal et al. [8]). The desired goal is to partition the graph by minimizing the weights of the edges across the partition.

Minimum cut: Given a graph, G = (V,E) with vertex (node) set V and edge (relation) set E, the minimum-cut algorithm tries to identify the smallest sum of edge weights that need to be removed to separate the graph V into two disconnected components for binary graph partitioning [64]. It has a complexity of $O(n^2)$, where n is the number of nodes (Karger [65]).

Ratio cut: Ratio cut is a measure of the quality of a partition. It is the ratio of the total edge weights between the clusters to the total edge weights within the clusters. The objective in ratio cut is to find a partition that minimizes this ratio, indicating a good separation of clusters. Ratio cut is often employed in the context of spectral clustering, especially for binary partitioning [66].

Multi-way graph partitioning: Multi-way graph partitioning is an NP-Hard problem, where the goal is to partition the set of vertices into k (greater than 2) clusters so that the weights of edges whose ends are in different partitions are minimized (Kernighan et al. [67]). The time complexity in this case increases exponentially with the value of k. A variation of this heuristic approach has been discussed by Fjällström [68].

Network-structure index: In this technique, the graph is partitioned into zones through a competitive flooding algorithm achieved through labelling seeds by zone identification, i.e. randomly selecting unlabeled neighbours and adding a label that matches its current value. The process repeats until all nodes are labelled [69].

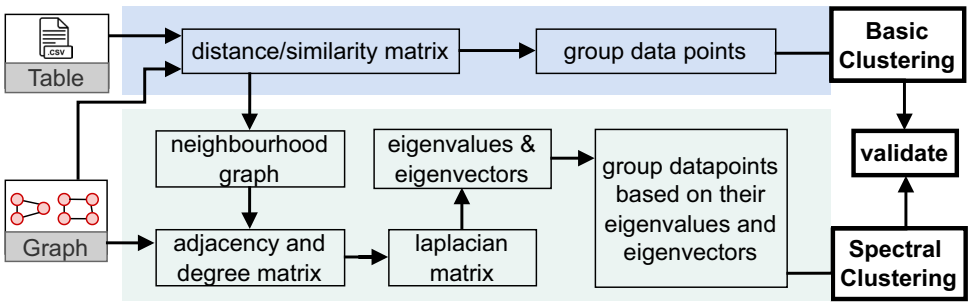

Fig. 5 Step-wise comparison of spectral algorithms against basic clustering algorithms. Dashed lines represent paths for spectral clustering while undashed lines represent paths for basic clusterings, such as K-means

Table 4 Comparison of spectral clustering algorithms discussed in this survey

Algorithm	Computational complexity	Laplacian	Application
EIGI [75]	$O(n^2)$ [11, 99, 100]	Un-normalized	Load balancing [101]
KP [78]	$O(n(bk^2 + bklog(n)))$ [11]		Parallel computing [102]
MELO [79]	$O(n^2d)$ [11]		VLSI design [103]
Hierarchical [90]	$O(n^3)$ [90]		General clustering [104]
Anchor [89]	$O(n^3)$ [13]	Non-Laplacian	
SpectralNet [91]	$< O(n^3)$ [94]		
Shi and Malik [80]	$O(n^3)$ [13]	Normalized	Text and document categorization [89]
Meila-Shi [82]	$O(n^3)$ [13]		Image segmentation [80]
NJW [17]	$O(n^3)$ [13, 107]		Dimension reduction [105]
KVV [83]	$O(n^3)$ [13]		Text clustering [106]
Self-tuning [85]	$O(n^3)$		General clustering
Co-trained multiview [86]	$O(tkn^2 + \sum_i^v vn(d^i)^2)$ [87]		
Constrained [88]	$O(n^2d)$ [88]		
Ultra-scalable [93]	$O(n(p^{\frac{1}{2}}d + K^2 + Kk + Kd + k^2t))$ [93]		
GNN for graph pooling [94]	$O(d(E + nd))$ [94]		
Quantum [95]	$O(T\frac{dlog(d)}{\delta^2})$ [95]	Projected normalized	General clustering [95]

n = number of nodes, d = number eigenvectors used, b = $max_{1 \leqslant i \leqslant n} d_i$, k = number of partitions (clusters), K = number of nearest representatives, p = number of representatives, t = Number of iterations, v = number of views, E = number of non-zero edges in coarsened Adjacency matrix, T = time of quantum state, δ = relative error

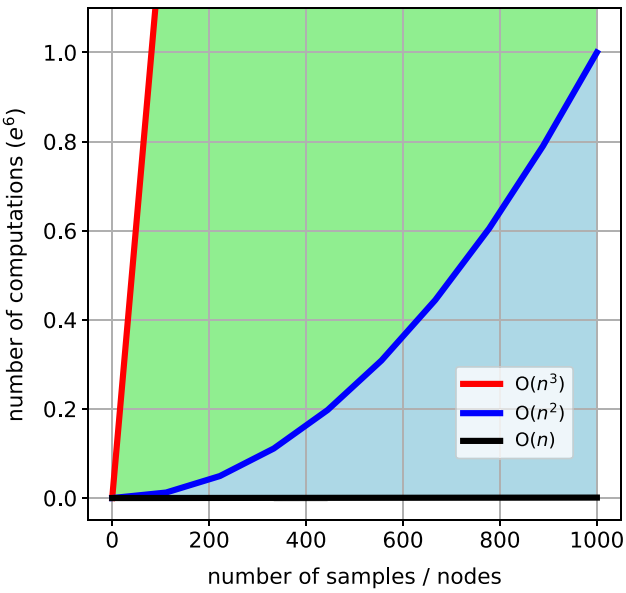

Fig. 6 Range of computational complexity of spectral clustering algorithms. Black line represents O(n), blue line represents $O(n^2)$ and red represents the highest possible complexity of $O(n^3)$. Values in the y-axis (number of computations) have raised to $1 * e^6$

Algorithm 4.3.1 EIGI algorithm [75]

1: **Input:** Graph G, its Laplacian matrix L and threshold r.
2: Calculate the second smallest set of eigenvalues and corresponding eigenvectors of L, using the Lanczos Algorithm.
3: Compare the second set of eigenvalues to threshold r and assign it to one of two clusters.
4: **Output:** Resulting partition.

Algorithm 4.3.2 KP Algorithm [78]

1: **Input:** Graph G, its Laplacian matrix L and the number of desired clusters k.
2: Find k eigenvectors of Laplacian, L and arrange them in matrix U.
3: Select k nodes to represent each k prototype
4: Calibrate k prototypes, iteratively, by calculating average and posterior selection to the closest node.
5: Verify whether cosine k prototypes with the connected points are higher than $\pi/8$ and reassign to larger cosine if so.
6: For all unallocated nodes, find the largest weight cut when compared to existing clusters, and assign them to clusters
7: **Output:** k partitions.

Girvan–Newman algorithm: A divisive clustering algorithm based on the concept of edge betweenness centrality (Girvan et al. [70]) which is the number of shortest paths passing through the endpoints of the edge. The algorithm starts with calculating edge betweenness for every edge in the graph, then removes the edge with the highest edge betweenness and calculates edge betweenness for the remaining edges. The process repeats until all edges are removed (Despalatović et al. [71]).

Determining quasi-cliques: While most partitioning algorithms try to minimize edge density, this technique focuses on maximizing edge density within a partition. To elaborate, a clique is a graph where all pairs of nodes have an edge between them and a quasi-clique is defined by imposing a lower bound on the degree of each vertex in the given set of nodes (Abello et al. [72]).

Min-hash approach: Min-hash approach attempts to define a node's outlinks (hyperlinks) as sets, i.e. two nodes are considered similar, if they share many outlinks [73]. The jaccard coefficient is used to represent the similarity between two nodes (Baharav et al. [74]).

4.3 Spectral clustering algorithms

Spectral clustering use eigenvalues and eigenvectors to represent clusters, as a set of vertices (nodes), derived from the node adjacency matrix of a graph [8]. These algorithms can provide lower/upper bounds for minimization/maximization of graph partitioning problems [11]. In the following, we discuss popular spectral clustering algorithms, along with their steps and complexity. In Fig. 5, we compare the steps of spectral clustering algorithms compared to a basic clustering algorithm, such as K-means. The steps are often repeated iteratively to reduce the value of a chosen cost function (SSE in K-means clustering). Spectral clustering algorithms have the additional steps of creating the Laplacian matrix and deriving eigenvectors and eigenvalues from it, which is why they have high computational costs, similar to hierarchical clustering (Table 3).

Algorithm 4.3.3 MELO algorithm [79]

1: **Input:** Graph G, its Laplacian matrix L, the number of desired clusters k and the number of eigenvectors to be used d.
2: Construct matrix of scaled eigenvectors.
3: Perform linear ordering on the eigenvectors.
4: Find the final k-way partition using linear ordering.
5: **Output:** k partitions.

Algorithm 4.3.4 SM algorithm [80]

1: **Input:** Graph G, its normalized Laplacian matrix L_{sym} and the number of desired clusters k.
2: Find k eigenvectors of the generalised eigensystem [81] and arrange them in matrix U.
3: Apply K-means algorithm on matrix U and find k partitions.
4: Assign nodes to clusters if their eigenvalue belongs to the partition.
5: **Output:** k partitions.

Algorithm 4.3.5 MS algorithm [82]

1: **Input:** Graph G, its normalized Laplacian matrix L_{rw} and the number of desired clusters k.
2: Find the k largest eigenvalues and eigenvectors of the generalized eigensystem [81] and arrange them in matrix U.
3: Apply K-means algorithm on matrix U in k-dimensional space
4: Assign nodes to clusters if their eigenvalue belongs to the partition.
5: **Output:** k partitions.

4.3.1 EIGI algorithm

The EIGI algorithm (see Algorithm 4.3.1) is based on the linear ordering of the Fiedler eigenvectors, performed using the Lanczos algorithm [11]. The eigenvector corresponding to the second smallest eigenvalue of a graph Laplacian matrix is usually referred to as the Fiedler vector, as defined by Doshi et al. [76]. The Lanczos method is an algorithm which is used to find a few extreme eigenvalues of a large symmetric matrix along with the associated eigenvectors (Parlett et al. [77]).

The Lanczos algorithm has a complexity of O(nk), where n = number of nodes and k = number of Lanczos iterations. As Nascimento [11] states, EIGI has the same computational complexity as the Lanczos algorithm, which would be $O(n^2)$ in the worst-case scenario, if k = n.

4.3.2 KP algorithm

The KP algorithm (Nascimento et al. [11])—defined from its k-way partitioning—intends to calculate how close nodes are by observing cosine similarities between pairs of rows from the eigenmatrix U (see Algorithm 4.3.2).

Algorithm 4.3.6 NJW algorithm [17]

1: **Input:** Graph G, its normalized Laplacian matrix L_{sym} and the number of desired clusters k.
2: Find k eigenvectors of the normalized Laplacian matrix L, arranging them in matrix U'.
3: Generate matrix U by normalising each row of U'.
4: Apply K-means algorithm on matrix U and find k partitions.
5: Assign nodes to clusters if their eigenvalue belongs to the partition.
6: **Output:** k partitions.

Algorithm 4.3.7 KVV algorithm [83]

1: **Input:** Graph G and the number of desired clusters k.
2: Find k eigenvectors of the generalised eigensystem [81], arranging them in matrix U.
3: Find k partitions using Cheeger Conductance.
4: Assign nodes to clusters if their eigenvalue belongs to the partition.
5: **Output:** k partitions.

Algorithm 4.3.8 Self-tuning spectral clustering algorithm [85]

1: **Input:** Graph G, its normalized Laplacian matrix L_{sym} calculated from optimal σ for every pair of nodes and desired number of clusters, k.
2: Find k eigenvectors of the normalized Laplacian matrix L, arranging them in matrix U'.
3: Generate matrix U by normalising each row of U'.
4: Apply K-means algorithm on matrix U and find k partitions.
5: Assign nodes to clusters if their eigenvalue belongs to the partition.
6: **Output:** k partitions.

4.3.3 Multiple Eigenvector linear orderings (MELO) algorithm

MELO algorithm (see Algorithm 4.3.3), is a greedy approach proposed by Alpert et al. [79], which partitions data into k segments using a dynamic programming procedure.

4.3.4 Shi and Malik (SM/KNSC) algorithm

A popular algorithm, Shi and Malik [80] (refer Algorithm 4.3.4) applied K-means algorithm on the eigenmatrix, where each row of the matrix is treated as a single object from the dataset.

4.3.5 Meila-Shi (MS/multicut) algorithm

Proposed by Meila and Shi [82] (refer Algorithm 4.3.5) the algorithm clusters a matrix of k largest eigenvalues. In this case, the normalized graph is formed through random walks.

Algorithm 4.3.9 Co-trained multi-view spectral clustering algorithm [86]

1: **Input:** Graph G, Laplacian matrices, e.g. L^1 and L^2 for two views (derived from similarity matrices S^1 and S^2), and the number of desired clusters k.
2: Get discriminative eigenvectors in each view U^1 and U^2.
3: Cluster U^1 and modify graph in view 2 and vice versa
4: Go to step 1 and repeat for a number of iterations.
5: **Output:** k partitions.

4.3.6 Ng–Jordan–Weiss (NJW/KNSC1) algorithm

The Ng–Jordan–Weiss algorithm proposed by Ng et al. [17] (refer Algorithm 4.3.6) is another improvement over the SM algorithm, given that the NJW algorithm applies K-means algorithm on a renormalized Laplacian matrix representing the dataset.

4.3.7 Kannan, Vempala and Vetta (KVV) algorithm

The KVV algorithm is an improvement over the SM algorithm with the KVV algorithm using Cheeger conductance for partitioning. Calculating the Cheeger conductance is beneficial in the context of graph partitioning and clustering because it provides a quantitative measure of the quality of a graph cut [84]. In order to find the Cheeger conductance or conductance of a cluster, the set of vertices is weighted to reflect their importance (Kannan et al. [83]).

4.3.8 Self-tuning spectral clustering algorithm

Most algorithms till now require the scaling parameter to be stated explicitly by the user, derived through domain knowledge, trial and error, or optimally found through several runs. To find the optimal hyperparameter value for scaling, for a given graph, Zelnik-Manor et al. [85] introduced a method to analyse the local scaling parameter σ for each data point. The self-tuning algorithm performs a similar eigendecomposition to NJW resulting in a worst possible complexity of $O(n^3)$

4.3.9 Co-trained multi-view spectral clustering algorithm

Multi-view data refers to data that is generated from different sources or observed from different perspectives (data pre-processing and/or analysis methods). As Yang et al. discussed [87], multi-view data refers to data objects that can be viewed from different angles or measured using different instruments, resulting in multiple views of the same data. Each individual view, in this case, has the possibility to lead to distinct knowledge discovery. These algorithms can be classified into five categories [87]:

- Co-training algorithms bootstrap clustering of the different views, either by using the prior or by gaining knowledge from one another.
- Multi-kernel learning predefine and combine kernels corresponding to each view, either linearly or non-linearly.
- Multi-view graph clustering fuses graphs from all views to a single graph and then implements graph-cut (e.g. node clustering) algorithms.
  - Multi-view spectral clustering
- Multi-view subspace clustering algorithms learn unified feature representations from all feature subspaces of all views.

Algorithm 4.3.10 Constrained spectral clustering algorithm [88]

1: **Input:** Graph G, its (Normalized) Laplacian matrix L_{sym}, threshold β, number of desired clusters K.
2: Calculate normalized constraint matrix $\overline{Q}$
3: Find the largest eigenvalue (λ_{max}) of $\overline{Q}$
4: **if** $\beta \geqslant \lambda_{K\text{-}1}.vol$ **then**
5: Return empty set of cluster assignment indicator u^*
6: **else**
7: Solve the generalized eigenvalue system;
8: Remove eigenvectors associated with negative eigenvalues and normalize the rest by, $v \leftarrow \frac{v}{\|v\|}\sqrt{vol}$;
9: $V^* \leftarrow \text{argmin}_{V \in \mathbb{R}^{N\times(K-1)}}$ trace $(V^T \overline{L} V)$, where the columns of V are a subset of the feasible eigenvectors generated in step 8;
10: $u^* \longleftarrow$ **K-means**$(D^{-\frac{1}{2}}V^*, K)$;
11: **end if**
12: **Output:** Cluster assignment indicator u^*, k partitions.

Algorithm 4.3.11 Anchor algorithm [89]

1: **Input:** Graph G and the number of desired clusters k.
2: Choose anchors at random, set k' = 1 and k" = 0 and assign final anchors as the farthest points from initially chosen points.
3: Construct clusters associated with anchors, x_k
4: Test if x_k has enough points, specified through threshold parameter.
5: Set k' = k' + 1. Choose x_k to be the farthest from all other existing anchors
6: If k' − k" < k, go to step 3
7: **Output:** k partitions.

Algorithm 4.3.12 Hierarchical spectral clustering algorithm [90]

1: **Input:** Graph G, its Laplacian matrix L, number of desired clusters k, indicated number of eigenvectors α.
2: Find the largest eigenvectors of L and produce the normalized feature vector space $T=(t_1,...,t_n)$
3: **for** $i \in 1, 2, \ldots, n$ **do**
4: $y_i = (t_{1,i}, t_{2,i}, \ldots, t_{\alpha,i})$
5: **end for**
6: Find k clusters by using a hierarchical algorithm with input y_i.
7: **Output:** k partitions.

- Multi-task multi-view clustering uses tasks to assess views and extracts inter-task knowledge to exploit multi-task and multi-view relationships.

In Algorithm 4.3.9, Kumar et al. [86] merge the co-training and the multi-view graph clustering as a novel approach to the problem of multi-view spectral clustering.

Algorithm 4.3.13 Spectral clustering using deep neural networks algorithm [91]

1: **Input:** Graph G from unlabelled data $X \subseteq R_{\mathrm{d}}$, loss of similarity $\mathrm{L}_{\mathrm{SpectralNet}}(\theta)$, Siamese net $\mathrm{L}_{\mathrm{siamese}}$ [92], number of desired clusters k and batch size m.
2: Construct a training set of positive and negative pairs and train a Siamese network.
3: Randomly initialize the network weights θ
4: **while** $\mathrm{L}_{\mathrm{SpectralNet}}(\theta)$ not converged **do**:
5: **Orthogonalization**:
a: Sample a random minibatch X of size m
b: Forward propagate X and compute inputs to orthogonalization layer Y'
c: Compute the Cholesky factorization LLT = Y'TY'
d: Set the weights of the orthogonalisation laser to be $\sqrt{m}(L^{-1})^{\mathrm{T}}$
6: **Gradient Step**:
a: Sample a random minibatch x_1,....,x_{m}
b: Compute the $m \times m$ affinity matrix W using the Siamese net
c: Forward propagate x_1.....,x_{m} to get y_1,....., y_{m}
d: Compute the loss
e: Use the gradient of $\mathrm{L}_{\mathrm{SpectralNet}}(\theta)$ to tune all Fθ weights, except for the output layer
7: **end while**
8: **Output:** Embeddings y_1,.....,y_{n}, $\mathrm{y}_{\mathrm{i}} \in \mathrm{R}_{\mathrm{k}}$, cluster assignments c_1, ...c_{n}, $\mathrm{c}_{\mathrm{i}} \in 1, \ldots \mathrm{k}$

4.3.10 Constrained spectral clustering algorithm

Constrained spectral clustering [88] (refer Algorithm 4.3.10) is a method of encoding many constraints into an algorithm such as K-means or hierarchical clustering. It uses the graph Laplacian and Eigenspace to explicitly encode ML (Must Link) and CL (Cannot Link) constraints to optimize the objective function for better results.

4.3.11 Anchor algorithm

Anchors hierarchy is a method of structuring data of generating nodes suited to the given task [89] (refer Algorithm 4.3.11). This concept has been used to define anchors for the anchor algorithm.

4.3.12 Hierarchical spectral clustering algorithm

HSC (Hierarchical based Spectral Clustering) [90] (refer Algorithm 4.3.12) is a novel clustering algorithm that combines spectral clustering with hierarchical methods to cluster datasets in convex and non-convex spaces more efficiently and accurately. It obviates the disadvantage of traditional spectral clustering by not using misleading information from noisy neighbouring data points, thus avoiding local optimum traps.

Algorithm 4.3.14 Ultra-scalable spectral clustering algorithm [93]

1: **Input:** Dataset $X = x_1, x_2, \ldots, x_N$
2: **Hybrid Representative Selection:**
a: Random sample of a set of p' candidate representatives such that $p < p' \ll N$.
b: p' candidates, we perform the K-means method to obtain p clusters and exploit the p cluster centres as the set of representatives.
c: Denote $R = r_1, r_2, \ldots, r_p$
3: **Fast Approximation Method of K-Nearest Representatives:**
Find the k-nearest representatives in order to construct a sparse affinity submatrix **B** that can be interpreted as a bipartite graph between objects and their respective **G={X, R, B}**
4: **Transfer Cut Utilisation:**
Apply the transfer cut method on the bipartite graph in order to partition it into multiple parts or "clusters" based on similarity metrics such as Euclidean distance or cosine similarity
5: **Output:** k partitions.

Algorithm 4.3.15 Spectral clustering with graph neural network for graph pooling [94]

1: **Input:** Graph G, its normalized Laplacian matrix, L_{sym}.
2: **Node Representation:** Use graph neural network (GNN) to compute node representations of L_{sym} as matrix, X.
3: **Cluster Assignment:** Use a multi-layer perception (MLP), with softmax activation to compute a soft cluster assignment as a matrix, S. This matrix represents the likelihood of nodes belonging to the different clusters.
4: **Optimization:** Optimize the parameters of GNN and MLP by minimizing an unsupervised loss function, $L_u = L_c + L_o$ which approximates the relaxed formulation of the mincut problem.
a: L_c is the cut loss term that evaluates the mincut based on the soft cluster assignment matrix S.
b: L_o is the orthogonality loss term that encourages cluster assignments to be orthogonal and clusters to be of similar size.
5: **Optimization:** Train the GNN and MLP jointly on the defined loss function, L_u using optimization techniques.
6: **Pooling and Graph Coarsening:** Use the soft cluster assignment matrix, S to perform pooling (downsampling) and generate a coarsened (reduced) version of the graph (MinCutPool layer).
7: **Hierarchical Coarsening:** Repeat steps 2 - 6 to obtain multiple layers of clustering
8: **Output:** k partitions.

Algorithm 4.3.16 Quantum spectral clustering [95]

1: **Input:** Graph G, its normalized Laplacian L and the number of desired clusters, k.
2: Calculate L projected on its k lowest eigenvectors, projected normalized Laplacian $\tilde{L}^{(k)}$.
3: Quantum clustering in the spectral space.
4: **Output:** k partitions.

4.3.13 Spectral clustering using deep neural networks algorithm

Spectral clustering using deep neural networks [91] (refer Algorithm 4.3.13) is a technique that uses deep neural networks to cluster data points into groups. It overcomes the limitations of scalability and generalization by training a network, called SpectralNet, which learns an embedding map from input data points to their associated graph Laplacian matrix and then clusters them.

4.3.14 Ultra-scalable spectral clustering algorithm

Ultra-scalable spectral clustering (U-SPEC) [93] (refer Algorithm 4.3.14) is an efficient algorithm for partitioning large datasets into clusters. It has nearly linear time and space complexity, allowing it to robustly and efficiently process 10-million-level nonlinearly separable data sets on a PC with 64 GB memory.

4.3.15 Spectral clustering with graph neural network for graph pooling

The algorithm (refer algorithm 4.3.15) employs Graph Neural Networks (GNNs) for spectral clustering, introducing a MinCutPool layer to coarsen the graph representation hierarchically. It utilizes a multi-layer perceptron (MLP) to compute soft cluster assignments based on node features, optimizing an unsupervised loss that balances cut loss and orthogonality loss. Through iterative pooling, the algorithm generates a hierarchy of coarsened graph representations, capturing diverse scales of structural information. End-to-end training ensures jointly optimized GNN and MLP parameters, demonstrating effectiveness in various tasks by avoiding degenerate solutions and handling imbalanced clusters.

4.3.16 Quantum spectral clustering algorithm

Of the several implementations of quantum spectral clustering algorithms [96–98], Kerenidis et al. [95] implemented a method for to group data with non-convex or nested structures. This method derives the normalized incidence matrix of a graph from the adjacency matrix to to calculate the Laplacian from. As a result the data is projected in a low-dimensional space where clustering can be done more efficiently and quickly than traditional methods using the spectral properties of the Laplacian matrix.

5 Discussion

5.1 Computational complexity of spectral clustering algorithms

Spectral clustering, at its worst, would provide a computational complexity of $O(n^3)$ to calculate the eigenvectors and eigenvalues from the adjacency matrix. This is similar to hierarchical clustering and some density-based approaches, e.g. OPTICS (Tables 3 and 4). However, spectral techniques benefit from the Laplacian representation of the data, which helps identify local neighbourhoods using eigenvectors. The least computationally expensive is the EIGI algorithm which employs the ratio cut solution to partition a graph into two clusters with a computational complexity of $O(n^2)$. The general range of the computational expenses of spectral clustering algorithms has been plotted in the Fig. 6.

The fast accelerating computational complexity against the increasing number of nodes and samples is one of the primary, if not the main, drawbacks of employing spectral clustering on large datasets. Scalability issues have been the key driving factor to investigate improved methods which lowers the the expense of spectral clustering algorithms down to $O(n^2)$ or even $O(n)$ [94].

The space (memory) complexity of spectral clustering algorithms are $O(n^2)$, at its worst, to store the square adjacency matrix and perform further calculations within the same memory storage. When compared with other clustering algorithms, spectral clustering algorithms are as computationally expensive as the agglomerative clustering algorithm. However, with spectral clustering, the benefits further outweigh the expenses as we get a representation of the data points in the eigenspace which can be used for other tasks than spectral clustering e.g. visualization.

5.2 Applications of spectral clustering algorithms

The primary strength of spectral clustering lies in partitioning a graph containing nodes, whether this graph is created from pixel data of an image, vectors generated from texts or documents or abundance data of proteins in samples. In Table 4, a comprehensive overview of the applications of different spectral clustering algorithms is provided. We can also observe a correlation between the type of laplacian matrix used and the application areas in this case.

The initial spectral clustering algorithms, where the unnormalized Laplacian matrices were used to generate eigenvalues and eigenvectors, mainly were used for tasks such as parallel computing, sparse matrix partitioning, electronic chip design (VLSI—Very Large Scale Integration). The introduction of the normalized Laplacian, in algorithms such as the SM, NJW, and self-tuning proved successful in image segmentation and general-purpose data analysis [108]. Some algorithms have been designed to handle very specific tasks such as the Anchor algorithm is used for text and document categorization. Additionally, there has been progress in the application of spectral clustering in several other domains such as protein abundance, gene expression, and social network analysis. Such progress deserves attention to motivate further research in graph data and spectral clustering.

5.3 Future research scope

As discussed in the previous section, scalability is still a major challenge that faces spectral clustering and as a result is one of the the major scope of improvement in the domain of graph clustering. While parallelization of calculations using GPUs are already created huge positive differences along with substantial decrease in computational complexity of algorithms, one issue that still persists is the recalculation of all intermediate steps when new data points are introduced for clustering on an existing model. Spectral clustering using deep neural networks and graph neural networks overcome this issue and possibly there could be solutions which uses simpler models than neural networks to solve this issue.

Another promising direction of improvement could be heterogeneous node clustering. While it is quite straightforward to create similarity graphs from homogeneous node attributes or features, it is quite challenging to create similarity graphs from heterogeneous set of nodes containing varying node attributes or features. This could be addressed by improved methods of meta-path selection, cross-domain generalization techniques and incorporating external information for heterogeneous set of nodes.

Acknowledgements The authors acknowledge the constructive feedback from Elias Kuiter that led to this survey. We also appreciate the German Research Foundation (DFG) for funding this project.

Author contributions Conceptualization: RM; methods: RM, EI; writing (original draft): RM; writing (review and editing): RM, EI, DW, RH, DB, GS; supervision: RH; funding: RH, DB, GS.

Funding Open Access funding enabled and organized by Projekt DEAL. This survey was funded by the German Research Foundation (DFG) under the project "Optimizing graph databases focusing on data processing and integration of machine learning for large clinical and biological datasets" (Grant Numbers: HE 8077/2-1, SA 465/53-1).

Declarations

Ethics approval and consent to participate Not applicable.

Consent for publication Not applicable.

Competing interests The authors declare no competing interests.

Data availability Not applicable.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Pokornỳ J. Graph databases: their power and limitations. In: IFIP international conference on computer information systems and industrial management. Springer; 2015. p. 58–69.
2. Miller JJ. Graph database applications and concepts with Neo4j. In: Proceedings of the southern association for information systems conference, Atlanta, GA, USA, vol. 2324. 2013.
3. Nurek M, Michalski R. Combining machine learning and social network analysis to reveal the organizational structures. Appl Sci. 2020;10(5):1699. https://doi.org/10.3390/app10051699.
4. Lee K, Barton D, Renson L. Modelling of physical systems with a hopf bifurcation using mechanistic models and machine learning. Mech Syst Signal Process. 2023;191: 110173. https://doi.org/10.1016/j.ymssp.2023.110173.
5. Mann M, Kumar C, Zeng WF, Strauss MT. Artificial intelligence for proteomics and biomarker discovery. Cell Syst. 2021;12(8):759–70. https://doi.org/10.1016/j.cels.2021.06.006.
6. Zhou J, Cui G, Hu S, Zhang Z, Yang C, Liu Z, Wang L, Li C, Sun M. Graph neural networks: a review of methods and applications. AI Open. 2020;1:57–81. https://doi.org/10.1016/j.aiopen.2021.01.001.
7. Ezugwu AE, Shukla AK, Agbaje MB, Oyelade ON, José-García A, Agushaka JO. Automatic clustering algorithms: a systematic review and bibliometric analysis of relevant literature. Neural Comput Appl. 2021;33(11):6247–306. https://doi.org/10.1007/s00521-020-05395-4.
8. Aggarwal CC, Wang H. A survey of clustering algorithms for graph data. In: Managing and mining graph data. Boston: Springer; 2010. p. 275–301. https://doi.org/10.1007/978-1-4419-6045-0_9.
9. von Luxburg U. A tutorial on spectral clustering. Stat Comput. 2007;17(4):395–416. https://doi.org/10.1007/s11222-007-9033-z.
10. Fernandes D, Bernardino J. Graph Databases Comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and OrientDB. In: Data. 2018;373–80.
11. Nascimento MC, De Carvalho AC. Spectral methods for graph clustering—a survey. Eur J Oper Res. 2011;211(2):221–31.
12. Donath WE, Hoffman AJ. Lower bounds for the partitioning of graphs. IBM J Res Dev. 1973;17(5):420–5. https://doi.org/10.1147/rd.175.0420.
13. Verma D, Meila M. A comparison of spectral clustering algorithms. Tech. rep. 2003.
14. Karim MR, Beyan O, Zappa A, Costa IG, Rebholz-Schuhmann D, Cochez M, Decker S. Deep learning-based clustering approaches for bioinformatics. Brief Bioinform. 2020;22(1):393–415. https://doi.org/10.1093/bib/bbz170.
15. Qi R, Ma A, Ma Q, Zou Q. Clustering and classification methods for single-cell RNA-sequencing data. Brief Bioinform. 2019;21(4):1196–208. https://doi.org/10.1093/bib/bbz062.
16. Inkaya T. A parameter-free similarity graph for spectral clustering. Expert Syst Appl. 2015;42(24):9489–98. https://doi.org/10.1016/j.eswa.2015.07.074.
17. Ng A, Jordan M, Weiss Y. On spectral clustering: analysis and an algorithm. In: Advances in neural information processing systems, vol. 14. 2001.
18. Kryszkiewicz M, Lasek P. TI-DBSCAN: Clustering with DBSCAN by Means of the Triangle Inequality. In: Rough sets and current trends in computing. 7th international conference. Berlin: Springer; 2010. p. 60–9.
19. Tarlow D, Swersky K, Charlin L, Sutskever I, Zemel R. Stochastic k-Neighborhood Selection for Supervised and Unsupervised Learning. In: Proceedings of the 30th international conference on machine learning, proceedings of machine learning research (PMLR, Atlanta, Georgia, USA), vol. 28. 2013. p. 199–207. https://proceedings.mlr.press/v28/tarlow13.html.
20. Mehta V, Bawa S, Singh J. Analytical review of clustering techniques and proximity measures. Artif Intell Rev. 2020;53(8):5995–6023.
21. Mohibullah M, Hossain MZ, Hasan M. Comparison of Euclidean distance function and Manhattan distance function using k-mediods. Int J Comput Sci Inf Secur. 2015;13(10):61.
22. Walters-Williams J, Li Y. Comparative Study of Distance Functions for Nearest Neighbors. In: Advanced techniques in computing sciences and software engineering. Dordrecht: Springer; 2010. p. 79–84.
23. Gultom S, Sriadhi S, Martiano M, Simarmata J. Comparison analysis of k-means and k-medoid with Ecluidience distance algorithm, Chanberra distance, and Chebyshev distance for big data clustering. IOP Conf Ser Mater Sci Eng. 2018;420: 012092. https://doi.org/10.1088/1757-899x/420/1/012092.
24. Benesty J, Chen J, Huang Y, Cohen I. Pearson correlation coefficient. Noise Reduct Speech Process. 2009. https://doi.org/10.1007/978-3-642-00296-0_5.
25. Kogge PM. Jaccard coefficients as a potential graph benchmark. In: 2016 IEEE international parallel and distributed processing symposium workshops (IPDPSW). 2016. https://doi.org/10.1109/ipdpsw.2016.208.
26. Shamir RR, Duchin Y, Kim J, Sapiro G, Harel N. Continuous dice coefficient: a method for evaluating probabilistic segmentations. 2018. https://doi.org/10.1101/306977.

27. Rahutomo F, Kitasuka T, Aritsugi M. Semantic cosine similarity. In: The 7th international student conference on advanced science and technology ICAST. 2012;4(1).
28. Currie D, Parry G. The impact of scallop dredging on a soft sediment community using multivariate techniques. Mem Qld Mus. 1994;36:315–26.
29. Hogben L. Spectral graph theory and the inverse eigenvalue of a graph. Electron J Linear Algebra. 2005;14:12–31.
30. Filippone M, Camastra F, Masulli F, Rovetta S. A survey of kernel and spectral methods for clustering. Pattern Recognit. 2008;41(1):176–90.
31. PalSingh R, Vandana V. Application of graph theory in computer science and engineering. Int J Comput Appl. 2014;104(1):10–3. https://doi.org/10.5120/18165-9025.
32. Angles R, Gutierrez C. Survey of graph database models. ACM Comput Surv. 2008;40(1):2. https://doi.org/10.1145/1322432.1322433.
33. Alm R, Imeri L. A performance comparison between graph databases: degree project about the comparison between Neo4j, GraphDB and OrientDB on different operations. 2021.
34. Hodler AE, Needham M. Graph Data Science Using Neo4j. In: Massive graph analytics. Boca Raton: Chapman and Hall/CRC; 2022. p. 433–57.
35. Mondal R, Do MD, Ahmed NU, Walke D, Micheel D, Broneske D, Saake G, Heyer R. Decision tree learning in neo4j on homogeneous and unconnected graph nodes from biological and clinical datasets. BMC Med Inform Decis Mak. 2022;22(6):1–13.
36. Vicknair C, Macias M, Zhao Z, Nan, X, Chen Y, Wilkins D. A comparison of a graph database and a relational database: a data provenance perspective. In: Proceedings of the 48th annual Southeast regional conference. ACM Press; 2010. https://doi.org/10.1145/1900008.1900067.
37. Khan W, Ahmad W, Luo B, Ahmed E. SQL Database with physical database tuning technique and NoSQL graph database comparisons. In: 2019 IEEE 3rd information technology, networking, electronic and automation control conference (ITNEC). IEEE; 2019. p. 110–6.
38. Sisodia D, Singh L, Sisodia S, Saxena K. Clustering techniques: a brief survey of different clustering algorithms. Int J Latest Trends Eng Technol. 2012;1(3):82–7.
39. Molchanov V, Linsen L. Overcoming the curse of dimensionality when clustering multivariate volume data. 2018.
40. Miller BA, Bliss NT, Wolfe PJ. Toward signal processing theory for graphs and non-Euclidean data. In: 2010 IEEE international conference on acoustics, speech and signal processing. IEEE; 2010. p. 5414–7.
41. Celebi ME. Partitional clustering algorithms. Cham: Springer; 2014.
42. Garima, Gulati H, Singh P. Clustering techniques in data mining: A comparison. In: 2015 2nd international conference on computing for sustainable global development (INDIACom). 2015. p. 410–5.
43. Ahmed M, Seraj R, Islam SMS. The k-means algorithm: a comprehensive survey and performance evaluation. Electronics. 2020;9(8):1295. https://doi.org/10.3390/electronics9081295.
44. Ankerst M, Breunig MM, Kriegel HP, Sander J. Optics: ordering points to identify the clustering structure. ACM SIGMOD Rec. 1999;28(2):49–60. https://doi.org/10.1145/304181.304187.
45. Fujita K. Approximate spectral clustering using both reference vectors and topology of the network generated by growing neural gas. PeerJ Comput Sci. 2021. https://doi.org/10.7717/peerj-cs.679.
46. Zhang P, Shen Q. Fuzzy c-means based coincidental link filtering in support of inferring social networks from spatiotemporal data streams. Soft Comput. 2018;22(21):7015–25. https://doi.org/10.1007/s00500-018-3363-y.
47. Manning CD, Raghavan P, Hinrich S. Hierarchical clustering. Cambridge: Cambridge University Press; 2019.
48. Miyamoto S, Ichihashi H, Honda K, Ichihashi H. Algorithms for fuzzy clustering. Berlin: Springer; 2008.
49. Grira N, Crucianu M, Boujemaa N. Unsupervised and semi-supervised clustering: a brief survey. Rev Mach Learn Tech Process Multimed Content. 2004;1:9–16.
50. Dempster AP, Laird NM, Rubin DB. Maximum likelihood from incomplete data via the EM algorithm. J R Stat Soc Ser B Methodol. 1977;39(1):1–38.
51. Ester M, Kriegel HP, Sander J, Xu X. et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: kdd. 1996;96:226–31.
52. Hinneburg A, Keim DA, et al. An efficient approach to clustering in large multimedia databases with noise, vol. 98. Konstanz: Bibliothek der Universität Konstanz; 1998.
53. Cao Y, Wu J. Projective art for clustering data sets in high dimensional spaces. Neural Netw. 2002;15(1):105–20.
54. Aggarwal CC, Wolf JL, Yu PS, Procopiuc C, Park JS. Fast algorithms for projected clustering. ACM SIGMoD Rec. 1999;28(2):61–72.
55. Agrawal R, Gehrke J, Gunopulos D, Raghavan P. Automatic subspace clustering of high dimensional data for data mining applications. In: Proceedings of the 1998 ACM SIGMOD international conference on management of data. 1998. p. 94–105.
56. Nagesh H, Goil S, Choudhary A. Mafia: efficient and scalable subspace clustering for very large data sets. Technical Report 9906-010; 1999.
57. Gan G, Ma C, Wu J. Data clustering: theory, algorithms, and applications. Philadelphia: Society for Industrial and Applied Mathematics; 2007. p. 183–298. https://doi.org/10.1137/1.9780898718348.
58. Barbará D, Li Y, Couto J. COOLCAT: an entropy-based algorithm for categorical clustering. In: Proceedings of the eleventh international conference on information and knowledge management. 2002. p. 582–9.
59. Bay SD, Pazzani MJ. Detecting change in categorical data: Mining contrast sets. In: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining. 1999. p. 302–6.
60. Holland J. Adaptation in Natural and Artificial Systems. Ann Arbor: University of Michigan Press; 1975.
61. Lee ML, Yang LH, Hsu W, Yang X. XClust: clustering XML schemas for effective integration. In: Proceedings of the eleventh international conference on information and knowledge management. 2002. p. 292–9.
62. Dalamagas T, Cheng T, Winkel KJ, Sellis T. Clustering XML documents using structural summaries. In: International conference on extending database technology. Springer; 2004. p. 547–56.
63. Aggarwal CC, Ta N, Wang J, Feng J, Zaki M. Xproj: a framework for projected structural clustering of xml documents. In: Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining. 2007. p. 46–55.
64. Ford LR Jr, Fulkerson DR. Flows in networks, vol. 54. Princeton: Princeton University Press; 2015.

65. Karger DR. Random sampling in cut, flow, and network design problems. Math Oper Res. 1999;24(2):383–413. https://doi.org/10.1287/moor.24.2.383.
66. Wei YC, Cheng CK. Towards efficient hierarchical designs by ratio cut partitioning. In: 1989 IEEE international conference on computer-aided design. digest of technical papers. IEEE; 1989. p. 298–301.
67. Kernighan BW, Lin S. An efficient heuristic procedure for partitioning graphs. Bell Syst Tech J. 1970;49(2):291–307. https://doi.org/10.1002/j.1538-7305.1970.tb01770.x.
68. Fjällström PO. Algorithms for graph partitioning: a survey. Linköping Electron Artic Comput Inf Sci. 1998;3(10).
69. Rattigan MJ. Maier M. Jensen D. Using structure indices for efficient approximation of network properties. In: Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining-KDD '06. 2006. https://doi.org/10.1145/1150402.1150443.
70. Girvan M, Newman MEJ. Community structure in social and biological networks. Proc Natl Acad Sci. 2002;99(12):7821–6. https://doi.org/10.1073/pnas.122653799.
71. Despalatović L, Vojković T, Vukićević D. Community structure in networks: Girvan-Newman algorithm improvement. In: 37th international convention on information and communication technology. electronics and microelectronics (MIPRO). 2014. p. 997–1002. https://doi.org/10.1109/MIPRO.2014.6859714.
72. Abello J, Resende MGC, Sudarsky S. Massive Quasi-Clique Detection. In: LATIN. 2002.
73. Aggarwal CC. Graph clustering. Boston: Springer; 2010. p. 459–67. https://doi.org/10.1007/978-0-387-30164-8_348.
74. Baharav TZ, Kamath GM, Tse DN, Shomorony I. Spectral Jaccard similarity: a new approach to estimating pairwise sequence alignments. Patterns. 2020;1(6): 100081. https://doi.org/10.1016/j.patter.2020.100081.
75. Hagen L, Kahng AB. New spectral methods for ratio cut partitioning and clustering. IEEE Trans Comput-Aided Design Integr Circuits Syst. 1992;11(9):1074–85.
76. Doshi V, Eun DY. Fiedler vector approximation via interacting random walks. Proc ACM Meas Anal Comput Syst. 2020;4(1):1–28.
77. Parlett BN, Scott DS. The Lanczos algorithm with selective orthogonalization. Math Comput. 1979;33(145):217–38.
78. Chan PK, Schlag MD, Zien JY. Spectral k-way ratio-cut partitioning and clustering. IEEE Trans Comput-Aided Design Integrat Circuits Syst. 1994;13(9):1088–96.
79. Alpert CJ, Kahng AB, Yao SZ. Spectral partitioning with multiple eigenvectors. Discret Appl Math. 1999;90(1–3):3–26.
80. Shi J, Malik J. Normalized cuts and image segmentation. IEEE Trans Pattern Anal Mach Intell. 2000;22(8):888–905.
81. Patanè G. Laplacian spectral basis functions. Comput Aided Geom Design. 2018;65:31–47.
82. Meilă M, Shi J. A random walks view of spectral segmentation. In: International workshop on artificial intelligence and statistics (PMLR). 2001. p. 203–8.
83. Kannan R, Vempala S, Vetta A. On clusterings: good, bad and spectral. J ACM. 2004;51(3):497–515.
84. Andoni, A. Lecture 11: Cheeger's Inequality and spectral graph partitioning. https://www.cs.columbia.edu/~andoni/advancedS20/scribes/scribe11.pdf.
85. Zelnik-manor L, Perona P. Self-Tuning Spectral Clustering. In: Advances in neural information processing systems, vol. 17. MIT Press; 2004. https://proceedings.neurips.cc/paper/2004/file/40173ea48d9567f1f393b20c855bb40b-Paper.pdf.
86. Kumar A, Daumé H. A co-training approach for multi-view spectral clustering. In: Proceedings of the 28th international conference on machine learning (ICML-11) (Citeseer). 2011. p. 393–400.
87. Yang Y, Wang H. Multi-view clustering: a survey. Big Data Min Anal. 2018;1(2):83–107. https://doi.org/10.26599/BDMA.2018.9020003.
88. Wang X, Qian B, Davidson I. On constrained spectral clustering and its applications. Data Min Knowl Discov. 2012;28(1):1–30. https://doi.org/10.1007/s10618-012-0291-9.
89. Moore AW. The anchors hierachy: using the triangle inequality to survive high dimensional data. CoRR abs/1301.3877. 2013. http://arxiv.org/abs/1301.3877.
90. Liu L, Chen X, Luo D, Lu Y, Xu G, Liu M. HSC: a spectral clustering algorithm combined with hierarchical method. Neural Netw World. 2013;23:499–521. https://doi.org/10.14311/NNW.2013.23.031.
91. Shaham U, Stanton K, Li H, Nadler B, Basri R, Kluger Y. Spectralnet: spectral clustering using deep neural networks. 2018. https://arxiv.org/abs/1801.01587.
92. Hadsell R, Chopra S, LeCun Y. Dimensionality eduction by learning an anvariant mapping. In: 2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06), vol. 2. 2006. p. 1735–42. https://doi.org/10.1109/CVPR.2006.100.
93. Huang D, Wang CD, Wu JS, Lai JH, Kwoh CK. Ultra-scalable spectral clustering and ensemble clustering. IEEE Trans Knowl Data Eng. 2020;32(6):1212–26. https://doi.org/10.1109/tkde.2019.2903410.
94. Bianchi FM, Grattarola D, Alippi C. Spectral clustering with graph neural networks for graph pooling. In: International conference on machine learning (PMLR). 2020. p. 874–83.
95. Kerenidis I, Landman J. Quantum spectral clustering. Phys Rev A. 2021;103: 042415. https://doi.org/10.1103/PhysRevA.103.042415.
96. Volya D, Mishra P. Quantum spectral clustering of mixed graphs. In: 2021 58th ACM/IEEE design automation conference (DAC). IEEE; 2021. p. 463–8.
97. Daskin A. Quantum spectral clustering through a biased phase estimation algorithm. TWMS J Appl Eng Math. 2017;10(1):24–33.
98. Gou S, Zhuang X, Jiao L. Quantum immune fast spectral clustering for SAR image segmentation. IEEE Geosci Remote Sens Lett. 2011;9(1):8–12.
99. Arora S, Hazan E, Kale S. Fast algorithms for approximate semidefinite programming using the multiplicative weights update method. In: 46th annual IEEE symposium on foundations of computer science (FOCS'05). IEEE; 2005. p. 339–48.
100. Golub GH, Van Loan CF. Matrix computations. Baltimore: JHU Press; 2013.
101. Van Driessche R, Roose D. An improved spectral bisection algorithm and its application to dynamic load balancing. Parallel Comput. 1995;21(1):29–48. https://doi.org/10.1016/0167-8191(94)00059-j.
102. Hendrickson B, Leland R. An improved spectral graph partitioning algorithm for mapping parallel computations. SIAM J Sci Comput. 1995;16(2):452–69. https://doi.org/10.1137/0916028.

103. Hagen L, Kahng A. New spectral methods for ratio cut partitioning and clustering. IEEE Trans Comput-Aided Design Integr Circuits Syst. 1992;11(9):1074–85. https://doi.org/10.1109/43.159993.
104. Alpert CJ, Yao SZ. Spectral partitioning. In: Proceedings of the 32nd ACM/IEEE conference on design automation conference—DAC'95. 1995. https://doi.org/10.1145/217474.217529.
105. Belkin M, Niyogi P. Laplacian eigenmaps and spectral techniques for embedding and clustering. Adv Neural Inf Process Syst. 2002;14:585–91. https://doi.org/10.7551/mitpress/1120.003.0080.
106. Dhillon IS. Co-clustering documents and words using bipartite spectral graph partitioning. In: Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data mining. 2001. https://doi.org/10.1145/502512.502550.
107. Gou S, Zhuang X, Zhu H, Yu T. Parallel sparse spectral clustering for SAR image segmentation. IEEE J Sel Top Appl Earth Obs Remote Sens. 2013;6(4):1949–63.
108. von Luxburg U, Belkin M, Bousquet O. Consistency of spectral clustering. Ann Stat. 2008;36(2):555–86. https://doi.org/10.1214/009053607000000640.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.