Multimedia Tools and Applications
https://doi.org/10.1007/s11042-022-13017-5

Graph partitioning and visualization in graph mining: a survey

Swati A. Bhavsar[1] · Varsha H. Patil[1] · Aboli H. Patil[1]

Received: 15 March 2021 / Revised: 17 July 2021 / Accepted: 28 March 2022

Abstract
Graph mining is a process of obtaining one or more sub-graphs and has been a very attractive research topic over the last two decades. It has found many practical applications dealing with real world problems in variety of domains like Social Network Analysis, Designing of Computer Networks, Study of Chemical Reactions, Bio Informatics, Program Flow Structures, Image Processing, Enterprise data, Sparse Matrix ordering and many more. For these applications, many graph classification and Graph Clustering algorithms are evolved. This paper presents a comprehensive survey of published work in Graph Mining by grouping them in a broad taxonomy. For each of these groups in the taxonomy, the basic concepts of the algorithms are covered in detail by mentioning the contributions of various authors to the basic concepts of each group. Furthermore, common issues in graph mining algorithms, such as clustering, partitioning, visualization of graphs, are also elaborated. Standard datasets available for graph mining are stated as well.

Keywords Graph mining · Graph classification · Graph partitioning · Graph datasets

1 Introduction

Graph mining is a process of obtaining required sub- graphs from a graph. A Graph containing vital information is further used for classification or clustering purpose. Graph mining has been

✉ Swati A. Bhavsar
bhavsar.swati4@gmail.com

Varsha H. Patil
varsha.patil@gmail.com

Aboli H. Patil
aboleee.patil@gmail.com

[1] Research Centre at Department of Computer Engineering, MCERC, SavitribaiPhule Pune University, Nashik, India

used for many different applications such as social network analysis, designing computer networks, chemical reactions, bio- informatics, program flow structures, image processing, enterprise data, chemical reactions, sparse matrix ordering [158].

To extract the information from graph data, the concept of graph mining was introduced [150]. Structure of data can be represented with a graph by considering the relationship among variables in the data. In the last decade, various researchers worked in Data Mining for enhancing performance and innovation. From the available research, it is found that an important characteristic of graph mining is structured data property in which the transactional graph mining settings are used. It considers databases of separate independent graphs. For example, in a molecule database, molecules are generally represented using atom and their bonding, an atom is treated as vertex and bonding between two atoms is represented as an edge. Whereas in a large network, data is represented with connected components. Some of the common examples of such networks include Internet, social networks, citation networks, concept networks, computer networks, chemical interaction networks, regulatory networks, socio-economic networks and encyclopedias [149] (Tables 1 and 2).

Over the past two decades, many research papers, books [27, 28, 37, 39, 149] and PhD thesis [15, 60, 74] have been written on Graph Mining. Several Survey papers [2, 61, 150] have been published on the said topic. Some of these surveys provide a good overview of graph mining algorithms. But only a limited number of methods provide details of graph clustering, graph classification, and sub-graph mining [99, 129].

The graph mining techniques are categorized into graph clustering, graph classification, and sub-graph mining.

- Graph clustering: In which, graph vertices are grouped together to form clusters based on the edge structure of the graph and some similarities of graph structures. Clustering is performed carefully such that the inter-cluster edges may be more but intra cluster edges should be less [129]. Clustering belongs to unsupervised learning technique, as the classes are not known in prior to clustering.
- Graph Classification: In which, the graph is classified into separate graphs [99]. This Classification is based on supervised or semi supervised learning technique where classes of the data are known in prior [99].
- Sub graph mining: In which, sub-graph is a graph whose vertices and edges are subsets of another graph. The frequent sub graph mining problem is used to produce the set of sub-

Table 1 Reported Applications of Graph Mining

Social Network Analysis	[4, 5, 9, 18, 21, 22, 27, 28, 37, 39, 48, 81, 87, 106, 112, 114, 116, 121, 133, 149, 150, 174, 178, 194]
Designing Computer Networks	[11, 45, 56, 67, 84, 99, 111, 129, 134]
Bio Informatics	[63, 79, 100, 105, 108, 125, 126, 128, 138, 147, 155, 180, 181, 182]
Program Flow Structures	[51, 92, 113, 156, 161]
Chemical Reactions	[10, 14, 34, 50, 73, 88, 153]
Sparse Matrix ordering	[17, 69, 90, 139, 143, 160, 187]
Image Processing	[16, 54, 75, 102, 117, 162, 115, 163, 184, 191]
Graph Matching	[42, 52, 59, 64, 110, 146, 165, 179, 192]
Sub-graph mining	[4, 52, 53, 123, 127, 137, 155]
Navigation System in GPS	[114, 118]

Table 2 Graph Mining Taxonomy

Types	Reported in
Graph clustering	[9, 13, 21, 22, 27, 30, 32, 37, 38, 39, 41, 71, 97, 101, 102, 104, 109, 113, 122, 131, 134, 149, 154, 157, 158, 175, 189, 194]
Graph classification	[4, 22, 23, 47, 63, 70, 96, 99, 168]
Sub-graph mining	[53, 79, 114, 123, 137, 155, 186]

graphs occurring in at least some given threshold of the given n input example graphs [99].

- Before moving towards more detailing, one needs to understand Basic Graph Theory. A graph G is a *set* $G = (V, E)$ where, V is a set of vertices, E is a set of edges of the graph and the number of vertices $n = |V|$ is the order of the graph. In an undirected graph, each edge is an unordered pair $\{v, w\}$. There are several terms related to the graph as discussed below-
- Connected Graph: A connected graph is a graph containing paths between all pairs of vertices. The graph is called disconnected, if the vertices cannot be reached from each other. The interconnection of graph is maintained through edge connectivity. Graph connected Graph G can be converted into disconnected Graph by removing the minimum number of edges is also referred as edge connectivity of graph.
- Cyclic Graph: If a graph contains a cycle means a simple path that begins and ends with the same vertex. And if a graph that contains no cycle then it is called as.
- Graph Sparsity / Density: Graph is called a sparse graph in which the number of edges is close to the minimal number of edges. Graph density of a graph in which the number of edges is close to the maximal number of edges.
- Static / Dynamic: Static graph consists of a fixed number of nodes and edges. While, in a Dynamic graph, insertion and deletion of edges and vertices can be performed at any time.
- Weighted / Unweighted Graph Distribution: It is a graph in which each branch is given a numerical weight. Therefore it is a special type of labeled graph in which labels are numbers, where an unweighted graph is a graph in which weights are not assigned to branches. Figure 1 shows the weighted graph. It has 6 vertices and 9 edges. Set of vertices is $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ and set of edges $E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9\}$, edge e1 connects vertices v_1 and v_2 whose weight is 5, edge e_2 connects vertices v_1 and v_3 whose weight is 6, edge e_3 connects vertices v_2 and v_3 whose weight is 8, edge e_4 connects vertices v_2 and v_4 whose weight is 3, edge e_5 connects vertices v_2 and v_5 whose weight is 4, edge $e_6$connects vertices v_3 and $v_5$whose weight is 6, edge $e_7$connects vertices v_4 and v_5 whose weight is 3, edge e_8 connects vertices v_5 and v_6 whose weight is 7, edge e_9 connects vertices v_4 and v_6 whose weight is 7.

- For all weighted graph, the degree of vertex v is addition of weighted of all edges incident on v. Figure 1 shows an example of weighted graph.
- Vertex Degree Distribution: Degree of a network is the number of connections and vertex degree distribution is probability distribution of these degrees.

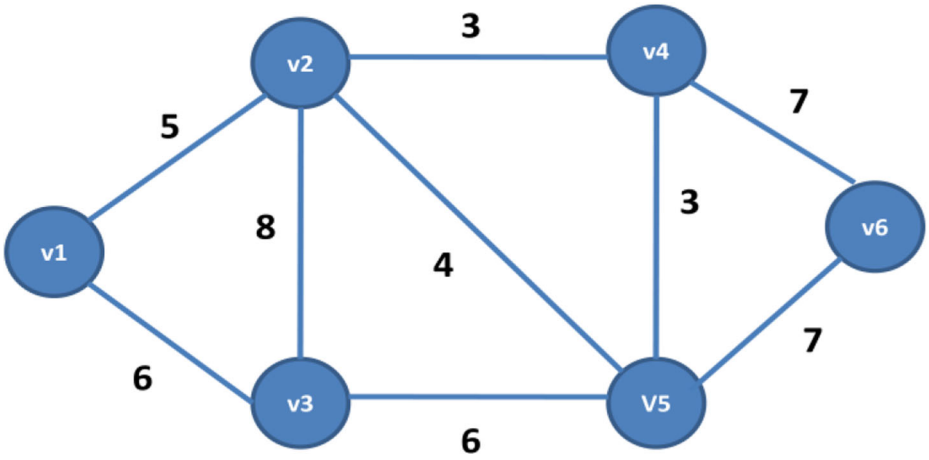

Fig. 1 Weighted Graph

- Directed / Undirected: For given graph $G = (V, E)$, the graph is undirected if $(v, w) \varepsilon E \Leftrightarrow (w, v) \varepsilon E$ graph as shown in the Fig. 2.1(a), otherwise the graph is directed if $(v, w) \varepsilon E \neq (w, v) \varepsilon E$. Directed graph is as shown in Fig. 2a and undirected graph is as shown in the Fig. 2b.
- Hyper-graph: Let H = (V, E), a hyper-graph is a generalization of a graph in which an edge can join any number of vertices. Formally, a hyper-graph is a pair where a set of elements is called nodes or vertices, and is a set of non-empty subsets of called hyper edges or edges (Fig. 3).

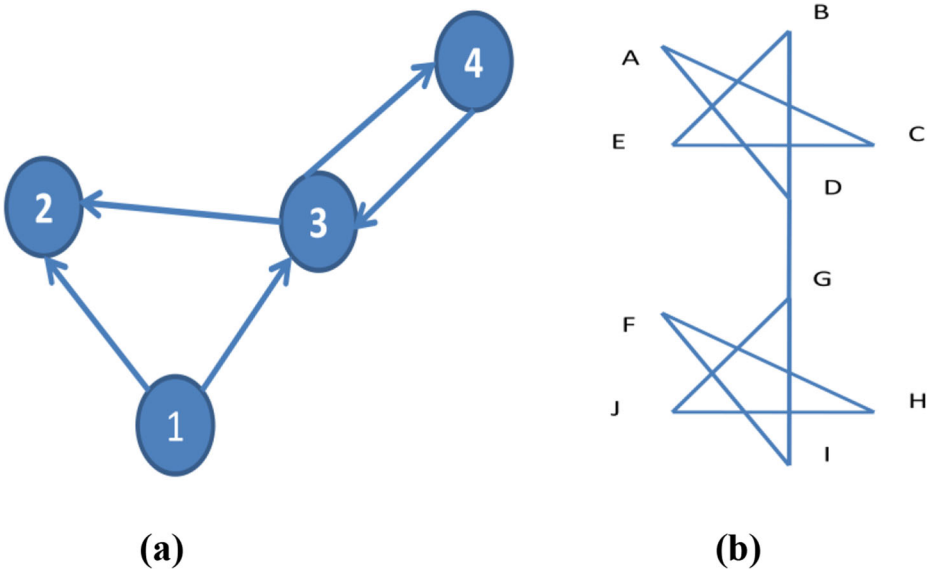

Fig. 2 **a** Directed Graph, **b** Undirected Graph [39]

$$v = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$$
$$e = \{e_1, e_2, e_3, e_4\} = \{\{v_1, v_2, v_3\}, \{v_2, v_3\}, \{v_3, v_5, v_6\}, \{v_4\}\}$$

- Adjacent Vertices and Edges, Incident Edges - Let G = (V, E) be an undirected graph. If e = (v, w) ∈ E, then v and w are said to be adjacent vertices of e and an edge e is incident on v, w. Two edges e1 and e2 are said to be adjacent if they are incident on the same vertex.
- Degree of vertex: Let G = (V, E) be an undirected graph and v ∈ V, degree of v is the number of edges incident to the vertex with loops counted twice.
- Loops and Parallel Edges: An edge is a loop if both ends of an edge are incident on the same vertex. If the same pair of vertices is adjacent by two or more edges, then those edges are called parallel edges
- Simple/ Multigraph: A graph G = (V, E) is simple if it does not have loops and/or more than one edge incident on the same vertex. Otherwise if more than one edge incident on the same vertex then the graph is known as a multi-graph.

Figure 4a shows a simple graph whereas Fig. 4b shows multiple graph with and edges e, and e4 are multiple graphs and edge e5 as self loop for vertex V2.

$$\deg(v) = \sum_{v,v' \in \mathbf{E}} w\left(v, v'\right) \tag{1}$$

Degree Matrix: Degree matrix is a diagonal matrix, which contains information about the degree of each vertex.

$$\mathrm{di,\,j} = \begin{cases} \deg(\mathrm{Vi}), & \text{if i == j} \\ 0, & \text{Otherwise} \end{cases} \tag{2}$$

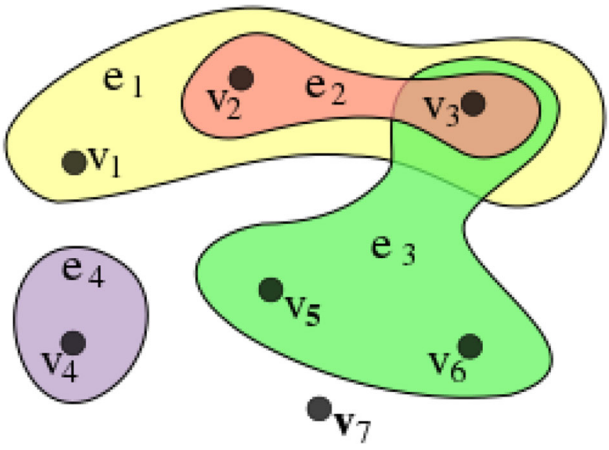

Fig. 3 Hyper graph [194]

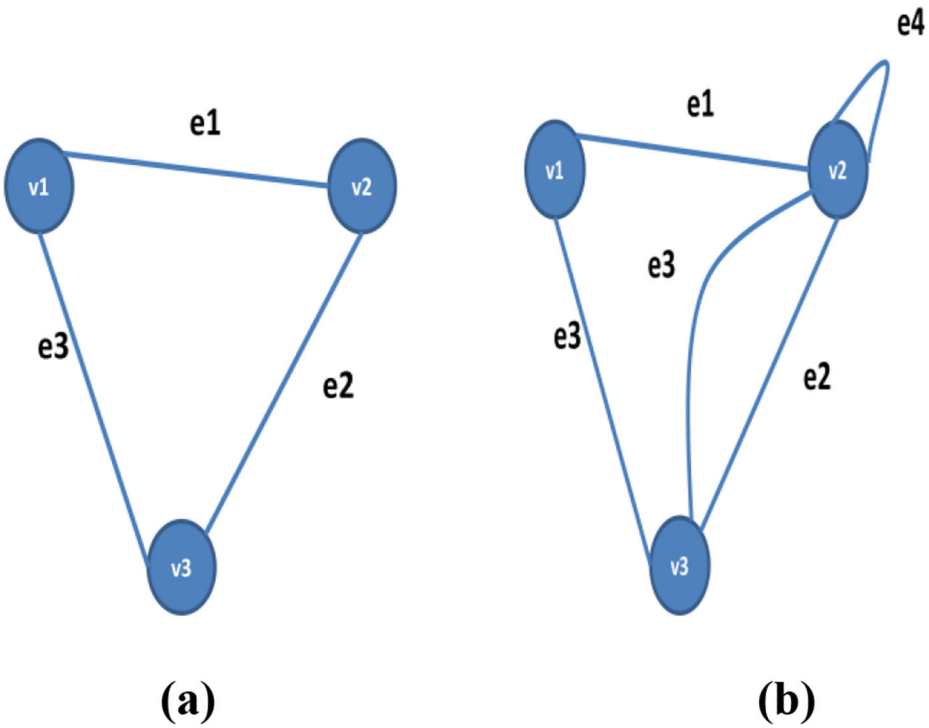

Fig. 4 **a** Simple Graph, **b** Multiple Graph

Adjacency Matrix: Let G = (V, E) be simple graph on v vertices then adjacency matrix A is of order v xv such that ∀ (i, j) ? {1,2,…v}

$$\mathrm{Ai,j} = \begin{cases} 0, & \text{if i == j} \\ \mathrm{w(i,j)}, & \text{Otherwise} \end{cases} \tag{3}$$

Figure 5 shows an Input graph, it's adjacency matrix representation and adjacency list representation of graph respectively.

- Laplacian Matrix: Let $G = (V, E)$ be a graph, then the matrix $L = D\text{-}A$ is called as a Laplacian matrix, where D is degree matrix and A is Adjacency matrix.
- Path and Circuit in Graph: Let $G = (V, E)$ be a graph and v1, v2 be nodes in a graph. If there is a sequence of edges from v1 to v2 then there is a path from v1 to v2. The path is called as a circuit the start and end vertex is the same.

Connected Graph: Let $G = (V, E)$ be a connected graph if there is a path between every pair of vertices.

- Matching of Graph: Let $G = (V, E)$ be a graph. Set containing pairwise nonadjacent edges is called as matching of the graph. Maximal matching includes the highest possible number of edges.

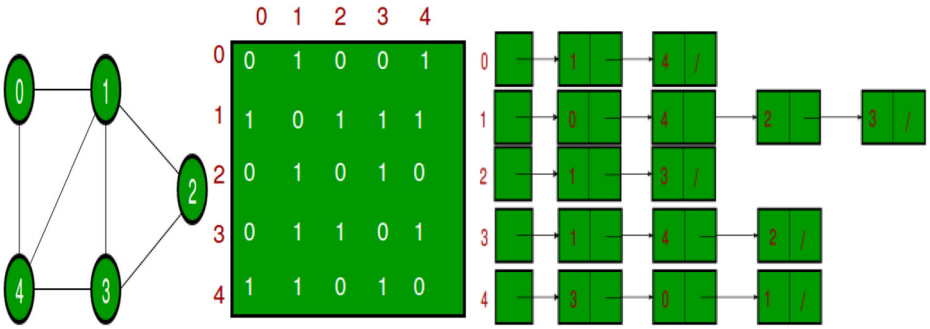

Fig. 5 Graph, Adjacency matrix of graph and Adjacency List of graph [186]

- Cut: A hyper-edge e of hyper-graph is known as a cut if, with respect to a particular partition, its vertices are place to more than one partition. The net cut of a partition is total number of hyper-edges that are cut. The weighted net cut of a partition is sum of weights of hyper-edges that are cut.
- Hyper-graph Partitioning: It is the process of finding a partitioning of hyper- graph in such a way that net cut cost function is minimized. It is also expected that a solution should satisfy the balancing constraints. Hyper-graph partitioning that results in two partitions is called bi-section.

Figure 6 shows proposed taxonomy for surveyed Graph Mining algorithms and their dedicated sections in this paper.

The rest of this paper is organized as follows. Section 2 provides taxonomy of graphs clustering algorithms, Section 3 provides details of graph classification, Section 4 reviews sub-graph mining methods, Section 5 discusses graph clustering approaches, Section 6 discusses various Graph layout and graph visualization techniques, and Section 7 is about assessment of graph mining algorithms, Section 8 deals with databases available for graph mining. Finally paper comes to a conclusion in section 10.

2 Taxonomy of graph mining

Graph mining algorithms are classified based on different types such as graph clustering, graph classification, and sub-graph mining.

Thus Graph mining helps for graph analysis and graph visualization. Various researchers have worked on these analysis and visualization techniques; some of them are summarized as shown in Table 3.

Figure 7 shows time line chart of graph clustering approaches proposed by various researchers and Fig. 8 shows time line chart of graph classification methods.

2.1 Motivation

As a promising technology, large graph mining is received a lot of attention in recent years. The concept of large graph mining is simple to understand and has two important parameters viz. graph analysis and graph visualization. It has a tremendous amount of applications in web graphs, social networks, recommendation systems, VLSI designing

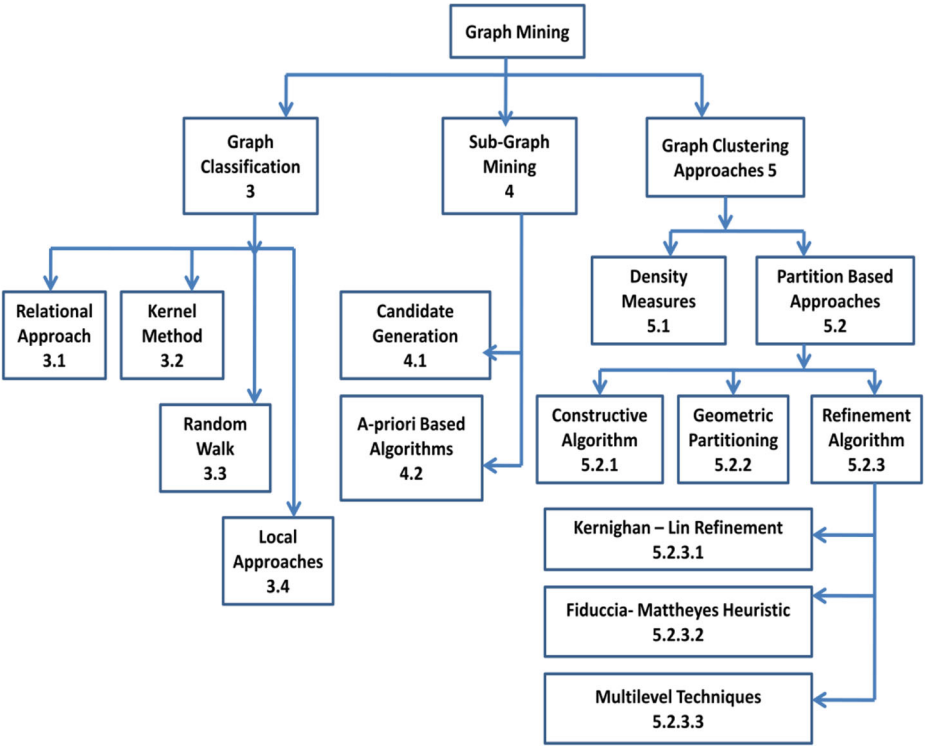

Fig. 6 Taxonomy of Graph Mining Algorithms

and many similar ones [185]. Literature reveals that noticeable graph partitioning and visualization algorithms were designed and developed by various researchers, but the combined study in the form of its applications, databases available in the graph mining area, techniques available for graph analysis, and visualization of large graph are not presented in a single paper. Hence it is a need of the day to present it in a single paper so it may act as a base paper for the researchers who are working in this area and attract attention of researchers to work in this area.

3 Graph classification

In general, graph classification is defined as the task of assigning class label a discrete class label set Y to input instances in an input space X. If molecular structure is considered then X = *{valid molecular structures}* and Y = *{toxic, non-toxic},* The graph structure as shown in Fig. 9 considered as toxic, then each full graph is assigned a class label and based on the label, classification is performed as toxic or non-toxic as shown in Fig. 9.

Table 3 Summary Of Graph Mining Techniques

Graph Analysis	[2, 4, 10, 13, 18, 19, 25, 37, 39, 73, 86, 89, 94, 144, 146, 149, 150, 158, 159, 166, 183]
Graph Visualization	[1, 6, 12, 13, 19, 26, 35, 36, 43, 44, 45, 47, 58, 66, 78, 85, 93, 95, 98, 108, 122, 130, 131, 151, 152, 154, 167, 170, 177, 188]

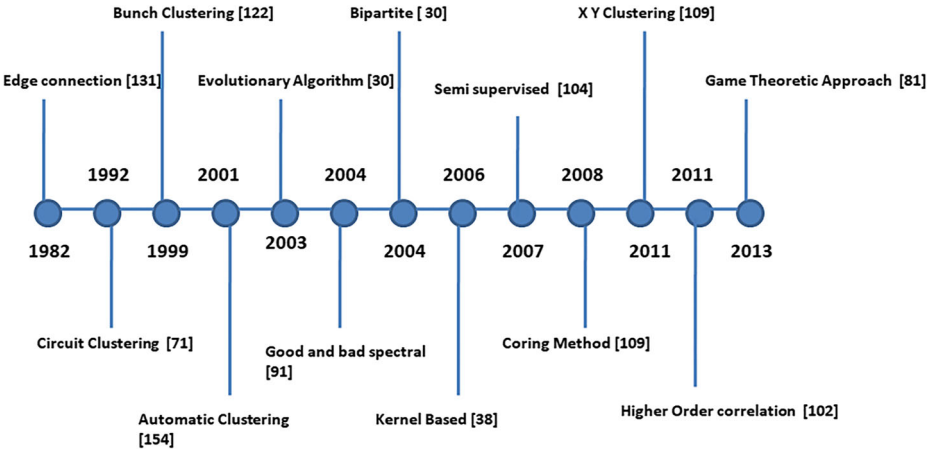

Fig. 7 Graph Clustering Approaches

The Classification approaches of the graph are classified into relational approaches, kernel method, random walks and local approaches [132].

3.1 Relational approaches

Almost all real-world data are stored by using the relational database; therefore, the relational approach is used to classify the objects in one relation. Multi-relational classification classifies an object using multiple relations. Multi-relational classification is used to discover useful patterns across multiple relations who are stored in TABLEs [25]. As a multi-relational approach is based on the traditional machine learning approach, it collects the random sample of homogeneous data from a single relation. As the real-world data is based on multi-relational and heterogeneous data. This solution cannot be realistically applied for it [176].

3.2 Kernel method

Kernel method is useful for graph classification. It is based on machine learning and data mining communities. Support Vector Machine(SVM) supports kernel method and uses kernel

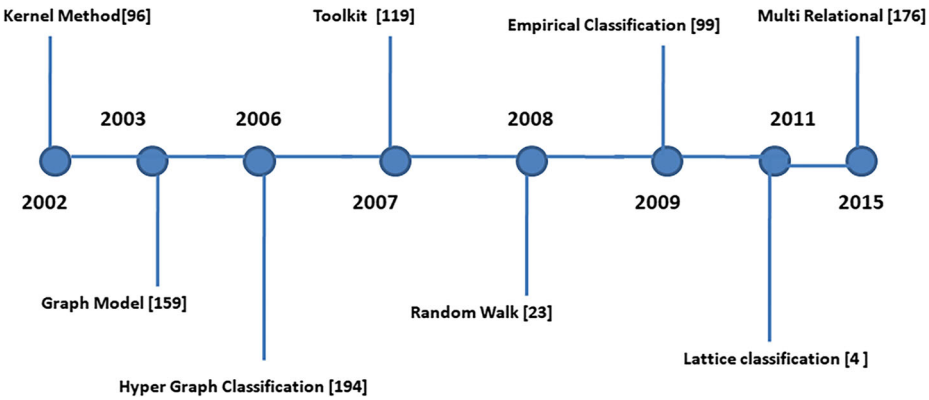

Fig. 8 Graph Classification methods

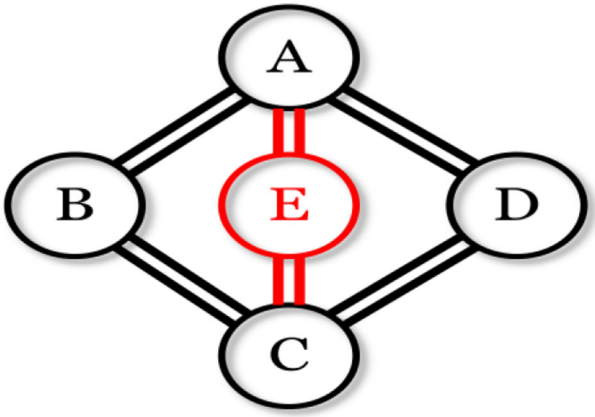

Fig. 9 Molecular graph with label as toxic [133]

function and eigen-values. Direct product kernel matrix is computed for all pairs of graphs. This matrix is used as input to SVM function to create the classification model. For creating kernel matrix, several assumptions are used as K is positive definite if and only if K has only non-negative eigen-values and K is strictly positive definite if and only if K has only positive eigen-values, that is no zero eigen-values [48]. Eigen values reflect positive definiteness of matrix A. A is said to be positive definite if it has all Eigen values that are non-negative. A is said to strictly positive definite if and only if it has only positive Eigen values.

Eigen values reflect positive definiteness of any matrix. *A* matrix *A* is said to be positive definite if it doesn't have any non-negative Eigen values. *A* is said to be strictly positive-definite if A has only positive eigen values. It means no zero Eigen values in it. Some properties of combinations of matrices are positive definite.

1. If there are two definite matrices A, B. Then product AB is positive definite.
2. For any strictly positive definite matrix B and integer n, Bn is strictly positive definite.

3.3 Random walk

Random Walk is based on the Markov Chain model. D^y is a random walk starting in a labeled node and ending when any node having the same labels which are reached for the first time. D-walk betweenness*(q, y)* is the expected passage time on an unlabeled node for each class [22]. The D walks have an ability to classify directed and undirected graphs. Its Complexity is linear with respect to -.

1. Number of edges in the graph.
2. Maximum walk length.
3. Number of classes.

This technique has been tested on the CORA database. The experimentation shows that it classifies the unlabeled nodes of the graphs and performs better than the techniques available

in the literature such as [47, 63]. Their main attainment is towards handling the graphs of large a number of nodes and edges.

3.4 Local approaches

Table 4 discusses details of local approaches used in graph mining, details of implementation and features of approaches.

4 Sub graph mining approaches

Frequent sub-graphs are beneficial for graph classification, clustering, and characterization of graph sets. As the sub-graph size decreases, the graph pattern size increases exponentially. It may lead to the issue of identification of sub-graphs and may also take more time. This issue can be solved by using scalable algorithms, which generate frequent sub-graphs in the search space [3, 7, 13, 63, 76, 91, 101, 135, 144]. The approaches used for sub-graph discovery are candidate generation and a- priori based algorithms.

4.1 Candidate generation

It is one of the important phases in frequent sub graph mining in which candidate sub-graphs are systematically generated. Some of the strategies, which identify candidate sub-graphs are listed below:

1. Level- wise Join: Two sub-graphs of size ‘m’ are combined together to form a (m + 1) candidate sub- graph.
2. Rightmost path extension:

Table 4 Local Approaches Used In Graph Mining

Paper	Technique/Method	Implementation	Features
[22]	D-Walks	CORA	Able to process large graphs
[103]	Multi-level Kernel k-means	Mutag, PTC	Able to find labels and edgesimilarity
[86]	Multi-level Kernel k-means	IMDB Movie	Running time efficiency is more
[40]	Genetic Algorithm	C++	Performance of Genetic Algorithm verified with various graph types
[193]	CFFfree	C++,VS	Works efficiently for larger graph having more nodes.
[97]	Coring Method	Micro Array dataset, image	Performs core region clustering
[32]	Clique matrix	D1MACS	Uses Clique matrix notation Clustering is also based on it.
	SSHGCA	MicroArraydataset	Uses background knowledge in clustering process
[159]	K-NN	Yahoo News, C++	More efficient and accurate for large size graph
[20]	HSG	PTE, DW_CM, Java	Association in graph determined
[53]	Distributed Algorithm	PTE, DW_CM, Java	More focus on efficient, distributed and heterogeneous graphs

In this candidate generation strategy, vertices are added on the rightmost path of m-sub tree to form (m + 1) sub tree. And other strategies such as extension and join, right-and-left tree join and equivalence class based extension are also used in candidate generation phase.

4.2 A- priori based algorithms

A- priori based approach is similar to frequent item-set mining and it works recursively [168]. Some of the a priori based frequent sub-graph mining algorithms are listed below.

– AGM [101]

This algorithm generates candidate graphs by merging any two candidate graphs. It checks whether resultant graph is a sub graph in a given graph. Two graph of size '*m'* are merged together to form a resultant graph of size *'m + 1'*. One of more resultant graphs of *size 'm + 1'* again fed to a- priori algorithm to obtain as a*'m + 2'*. As two graphs are merged together, graph size is increased by one vertex, hence it is called as vertex based candidate generation algorithm.

– FSG[91]

This method is based on edge based candidate generation process. The number of edges in a graph represents size of graph. Similar to vertex based candidate generation method, two graphs of size *'m'* are merged together to form resultant graph of size '*m + 1'* which must be frequent. In further iterations, it generates candidate sub graph whose size is exactly greater than 1 of previous ones.

– Edge disjoint path-join algorithm[144]

It measures the number of disjoint paths of a graph. A path is said to be disjoint if they do not share a common edge. Two candidate graphs of 'm' disjoint paths are merged together to form a resultant graph containing *'m + 1'* disjoint paths.

5 Graph clustering approaches

The vertices of the graph are grouped together into clusters by considering the edge structure of the graph. Graph is divided into clusters by considering the important properties like cluster fitness measures. This function is used to decide the quality of cluster. Following measurement are used for deciding cluster fitness measure.

5.1 Density measures

In this technique, density of a graph is computed based on threshold value and based on the threshold value, maximal sub-graphs are searched [67].

5.2 Partitioning based approaches

In addition to direct density measures, connectivity measures with the rest of the graph are used to identify high-quality clusters. Measures of the "independence" of a sub-graph of the rest of the graph have been defined based on cut sizes. Possibly the most important such measure in the context of clustering is conductance [52]. It leads to graph partitioning. The problem is to partition the vertices of a graph into equal parts such that the number of cuts should be minimized. A graph-partitioning problem is NP complete. Many algorithms are invented to achieve good partitioning. Graph Partitioning algorithms are mainly classified into constructive algorithms, geometric algorithms, Refinement algorithms and Multi level Methods.

5.2.1 Constructive algorithm

Constructive algorithm is a type of heuristic method [56, 72]. It starts with an empty solution and repeatedly extends the current solution until a complete solution is obtained Examples of some constructive heuristics developed for famous problems are: flow shop scheduling and vehicle routing problem.

Exact Algorithm Applying branch and bound techniques can solve the GPP problem. Bounds are determined by using semi definite programming [87], multi commodity flow [99] and linear programming [166].

Hanger et al. applied branch and bound technique to Graph Partitioning Problem in the continuous quadratic form [137]. Furthermore, Dellinget. al had derived bounds by computing minimum s-t cut between vertices *(v1, v2)* such that $V1 \cap V2 = \varnothing$ [96]. This method is useful for the partitioning of complex road network having millions of nodes. But the limitation of this method is running time is more and running time is dependent on bisection width of the graph.

All these methods are used in solving small size graph partitioning problem and having large running time. Modified Lanczo's algorithm can be used to computer Eigen vector with respect to second smallest Eigen value but it has high running time. This method produces good partitions but the drawback is it can partition graph only into 2 k blocks.

Random Partition This method of partitioning is applied to small graphs, such as coarsened graphs, which are used in multilevel partitioning. This algorithm is easy for implementation and its time complexity is *O(n)*. But it produces a poor solution towards finding edge cut value. The number of edge cuts produced is far more than that of best-known cut.

Spectral bisection method It is one of the excellent partitioning techniques used to divide the graph into sub graphs [168]. In this method, global information about associability of a graph is maintained by computing Laplacian matrix with its Eigen values. The steps of the algorithm are listed in algorithm 1.

Algorithm 1: Spectral bisection.

For every Graph G= {V, E}

1. Represent Adjacency Matrix A_{ij}
2. Compute Degree Matrix D
3. Compute Laplacian Matrix $L = D - A_{ij}$
4. Eigen values(λ) of L gives a lower bound of optimal cost© of partition
5. Eigen Vector (V) corresponding to λ, called as Fiedler Vector.
6. Determine Median of V
7. For each node u_1 of G if u_i
8. It bisects the graph into two sub graphs based on the sign of corresponding vector entry.
9. Set up graph $G_1 = (V_1, E_1)$and $G_2 = (V_2, E_2)$

Greedy Construction Greedy construction algorithm is one of the constructive heuristic algorithms. The detailed algorithm is explained in algorithm 2.

Algorithm 2: Greedy Constructive procedure

Input: Graph $G = (V, E)$ desired size k of partition
Output: k – partition of V

1. $P \leftarrow \{P_0, \ldots P_{k-1}\}$
2. $V' \leftarrow$ V
3. for$p \in [0, k-1]$ do
4. $u \leftarrow$ random vertex from V'
5. $P_p \leftarrow \{u\}$
6. $V' \leftarrow V' \setminus \{u\}$
7. $p \leftarrow 0$
8. While $|v'| > 0$ do
9. $b \leftarrow$greedy_function(V', P, p, G)
10. $P_p \leftarrow P_p \cup \{b\}$
11. $V' \leftarrow V' \setminus \{b\}$
12. $P \leftarrow (p+1) \bmod k$
13. Return P;

Initially, all partitions are empty, k different random vertices are also known as seed vertices are assigned to those empty partitions. Remaining vertices are greedily assigned to each partition in a circular fashion. Total n-k iterations are performed. Greedy function is used to build kway initial partition [97].

This algorithm is specifically based on the selection of seeds. If bad seeds are chosen, then leads to a problem of poor quality in terms of finding number of edge cut value.

Greedy Graph Growing Partition algorithm This algorithm minimizes total gain at each vertex. Vertices are assigned in a circular order of part and grow as a whole iteratively [97].

The algorithm structure is same as that of Algorithm 1 and greedy function is added as shown in algorithm 2.

Algorithm 2: Greedy Graph Growing Partition Algorithm

Input: A set of vertices V' of unassigned vertices, the current partition P, partition index p and Graph G = (V, E, c, w)
Output: A vertex assigned in P_p
1. m← min $_{v \in V'}$(tp (v, p))
2. C←{ $_{v \in V'}$|tp (v, p)=m}
3. return random vertex from C

This algorithm requires fewer iterations as compare to GGP algorithm, but it suffers from the same problem of poor quality partitions with chosen of bad seeds.

5.2.2 Geometric partitioning

Another approach for graph partitioning is Geometric Partitioning [55, 156]. This algorithm produces partitions, which are based on bounds; exist for special classes of the graph. It can compute the partition faster than spectral bisection method. But the partitions obtained are worse than spectral bisection method. It requires multiple trails toward getting the solution, so it increases the time requirement. But the overall runtime is less than the time required for spectral methods. Geometric graph partitioning has limited application areas because geometric information is not available in every case linear programming where geometry is not associated with graph. An algorithm has been developed [142] to compute geometry information of graph. This method is based on computing spectral information. It is expensive and dominates overall time taken by graph partitioning algorithms.

5.2.3 Refinement algorithm

A balanced bipartition $\{A, B\}$ can be achieved by swapping subsets $X \varepsilon A$ and $Y \varepsilon B$ with $|X| = |Y|$ to opposite sides so that cut size is reduced. One of the best refinement algorithms is Kernighan Lin Algorithm [31]. Furthermore, it is replaced with Fiduccia and Mattheyeses algorithm [177]. It uses bucket data structure leads to reduce time complexity from $O(n^2)$ from $O(m)$.

Kernighan Lin Refinement Algorithm Kernighan and Lin proposed KL- local search method to perform graph partitioning. This algorithm is iterative in nature. The input to the algorithm is an undirected graph $G = (V, E)$ with vertex set V, edge set E, and may have numerical weights on the edges in E. The goal of the algorithm is to partition V into two disjoint subsets X and Y of equal (or nearly equal) size, in a way that minimizes the sum T of the weights of the subset of edges that cross from X to Y. If the graph is unweighted, then instead the goal is to minimize the number of crossing edges; this is equivalent to assigning weight one to each edge [62]. The algorithm maintains and improves a partition, in each pass using a greedy algorithm to pair up vertices of A with vertices of B, so that moving the paired vertices from one side of the partition to the other will improve the partition. is used to achieve two subsets of vertices of

equal size. Swapping of vertices in two different subsets helps to decrease the edge cuts obtained. It starts with any partition of *V (G)* into *X* and *Y*. If the graph is a weighted graph, then its weight is considered and if it is an unweighted graph, then the unique weight is considered.

- Internal Cost**:** It is the cost of edges connecting node v_i within its own group. Suppose Internal cost of vertex $v_i \in X$ is the addition of edge weights of adjacent vertices of v_i in same partition that is *X* only. Likewise, for each node, the internal cost is computed.

$$I(Vi) = \sum_{vj \in \mathbf{X}} w(vi, vj) \tag{4}$$

- External Cost- It is the cost of edges connecting node vi within another group. Suppose Internal cost of vertex vi ∈ X is the addition of edge weights of adjacent vertices of vi in other partition that is Y only. Likewise for each node, internal cost is computed.

$$E(V_i) = \sum_{v_j \in B} w(v_i, v_j) \tag{5}$$

– The variation of external cost and Internal cost is computed as

$$G(v_i) = E(v_i) - I(v_i) \tag{6}$$

– Compute difference of External cost and internal cost for each vertex where vi ϵ V
– If vertices v1 and v2 are exchanged then their gain are computed by using formula

$$g\,(v1, v2) = G\,(v1) + G(v2) - 2w(v1, v2) \tag{7}$$

– swap the pair of vertices for which gain value is maximum
– if v1 and v2 are interchanged, then mark them as locked vertices
– for all such unlocked vertices from both divided sets excluding v1 and v2, update the gain values by formula

$$g'(vi) = g(vi) + 2w(vi, v1) - 2w(vi, v1), \forall vi \in X - \{v1\} \quad (8)$$

$$g'(vj) = g(vj) + 2w(vi, v2) - 2w(vi, v2), \forall vi \in X - \{v2\} \quad (9)$$

- Repeat above procedure, if no more vertices are to swap.
- If initial partition is good, then KL algorithm is more efficient and faster. The complexity of this algorithm is O (|E|2log|E|).

Algorithm 3: A Heuristic Kernighan Lin Procedure

```
Function Kernighan-Lin (G (V, E)):
1.  Determine a balanced initial partition of the nodes into sets X and Y
2.  do
3.  compute G values for all v1 in X and v2 in Y
4.      Let gv, av and bv are empty lists
5.  for (n := 1 to |V|/2)
6.          find V1 from X and V2 from Y, such that g = G[v1] + G[v2] - 2* W(v1, v2) is
            maximal
7.  remove v1 and v2 from further consideration in this pass
8.  add g to gv, a to av, and b to bv
9.  update G values for the elements of X = X \ v1 and Y= Y \ v2
10. end for
11. find k which maximizes g_max, the sum of gv[1],...,gv[k]
12. if (g_max> 0)then
13.          Exchange av[1],av[2],...,av[k] with bv[1],bv[2],...,bv[k]
14. until(g_max<= 0)
15. return G(V,E)
```

The limitations of Kernighan Lin algorithm are in terms of time complexity and cut size. Time complexity is $O(n^2)$ and unable to minimize cut size for imbalance structure of the graph. These parameters improvement is proposed by Fiduccia – Mattheyses (FM) heuristic algorithm in which time complexity is reduced to $O(m)$ and also deals with imbalanced partition by moving single vertex instead of swapping two of them.

Fiduccia- Mattheyes Heuristic The Fiduccia-Mattheyses (FM) heuristic is beneficial for hyper graph bi-partitioning. This is an iterative algorithm type. It aims at reducing net-cut costs. It works on the same principle of KL Heuristic. It also starts with random solution and sequence of moves is organized as passes. Only a single vertex is moved across the cut in a single move. It can handle unbalanced partitions, for most efficiently handling a balance factor is introduced. A special data structure known as bucket is used to select vertices to be moved across the cut to improve running time. All each possible move is with its change in cost and known as gain. A move with highest gain is selected

and execute. For the selected with highest gain, it is locked with consideration to the value of r. After successful execution of move vertex, it leads to change in gains of adjacent vertices, and hence their gains are updated. Selection and execution of best gain move, gain update are repeated until every vertex is locked. Then best solution seen during the pass is adopted as starting solution of next pass. The algorithm terminates when a pass fails to improve solution quality. All moves with the same gain are stored in a linked list representing a gain bucket [77].

Algorithm 4: Heuristic Fiduccia- Mattheyes Procedure.

```
Input: A hyper graph with a vertex (cell) set and a hyper edge (net) set
Output: 2 partitions
1.                          gaincontainer.unlock all();
2.  n(i): # of cells in Net i; e.g., n(1) = 4
3.  s(i): size of Cell i
4.  p(i): # of pins of Cell i; e.g., p(1) = 4
5.  C: total # of cells; e.g., C = 13
6.  N: total # of nets; e.g., N= 4
7.  P: total # of pins; P= p(1) + … + p(C) = n(1) + … + n(N)
8.  Area ratio r, 0< r<1
9.                          Cutsetsize is minimized by FMpass (gain container,
partitionment)
10.                         solution cost = partitionment.get cost();
11.                         while(not all vertices locked)
12.                         move = choose move();
13.                         solution cost += gain container.get gain(move);
14.                         gaincontainer.lock vertex( move.vertex() );
15.                         gain update(move);
16.                         partitionment.apply(move);
17.                         roll back partitionment to best seen solution;
```

Multilevel Technique Bernad et al. have introduced multilevel graph partitioning method. This method helps in accelerating the existing graph partitioning tool. The main idea behind this algorithm is to group vertices together in order to deal with groups of vertices instead of processing independent vertices in case of the partitioning of a larger graph in k parts. A multilevel method is divided into following three steps in consideration with Consider a weighted graph $G_0 = (V_0, E_0)$.

Coarsening Phase The graph G_0 is reduced into a sequence of smaller graphs $G_1, G_2 \ldots G_m$ such that $|V_0| > |V_1| > |V_2| > \cdots > |V_m|$. Random matching method is one, which is used for collapsing of vertices and to form a multi-node.

Partitioning Phase A 2-way partition P_m of the graph $G_m = (V_m, E_m)$ is computed that partitions V_m into two parts, each containing half the vertices of G_0.

Uncoarsening Phase The partition P_m of G_m is projected back to G_0 by going through intermediate partitions $P_{m-1}, P_{m-2} \ldots P_1, P_0$.

Multilevel graph partitioning is shown in Fig. 10. Coarsening deals with providing a global outline of a graph. Partitioning phase deals with the partitioning of the original graph. Uncoarsening phase improves partition by preserving the fine structure of partition.

Graph Coarsening It is an iterative method. Suppose Graph G_i has a setoff vertices V_i are grouped together to form a single vertex v for the next level graph G_{i+1}. Vertex v is known as multi node. V_i^v is the set of vertices of G_i which are grouped together to form v of G_{i+1}

$$W(v) = \sum\left(V_i^v\right) \tag{10}$$

The cut value of partitions in coarsen graph is same as that of cut value for the partition in original graph. Graph Coarsening can be performed by two ways. One is known as random matching and clubbing of matched vertices into a multimode. Another approach is based on clustering phenomenon in which highly connected vertices generates multimode (Figs. 11, 12 and 13).

Matching is defined as a set of edges, in which no two edges are incident on the same vertex. Graph G_i *is transformed into* G_{i+1} by coarsening phase in which vertices to be matched are collapsed together to form a multi-node. When the vertices are collapsed, the size

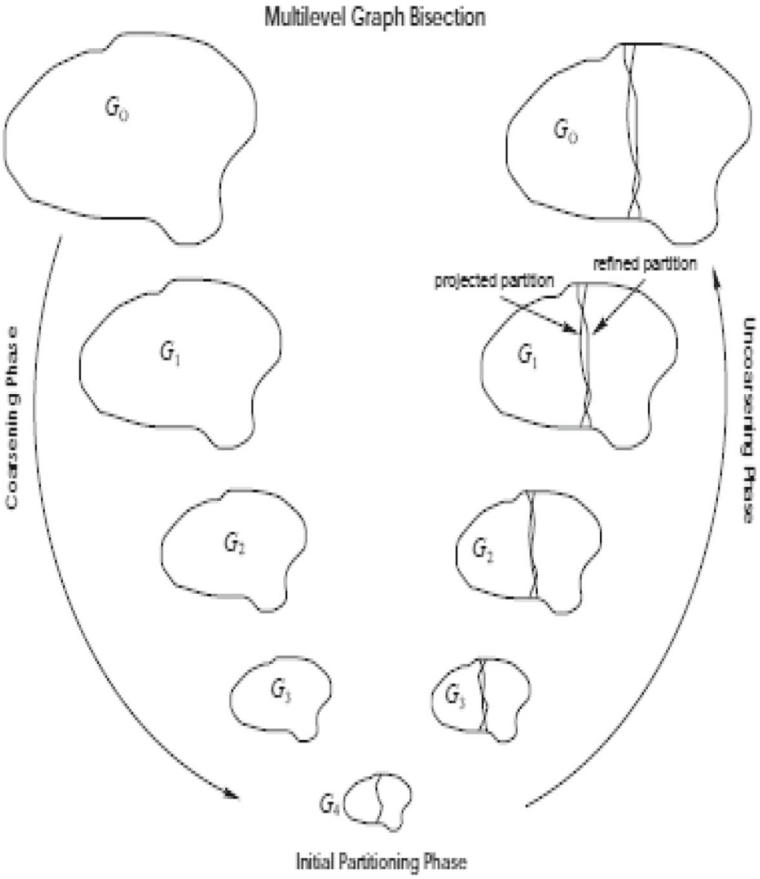

Fig. 10 Multilevel k – Partitioning [99]

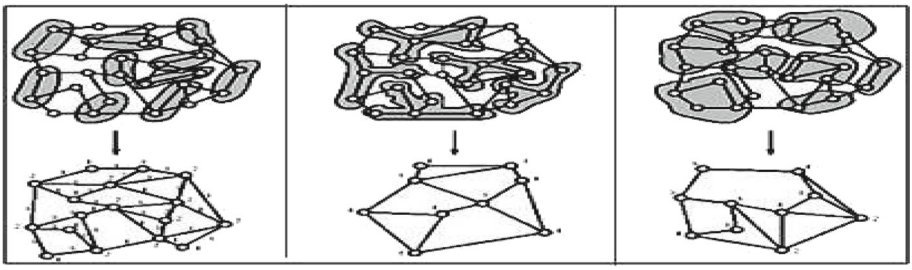

Fig. 11 Graph Coarsening Examples [120]

of the original graph is reduced. If the matching is of maximum size, then it is called as the maximal matching. The size of maximal matching is different based on the matching's calculated. This maximal matching is useful in coarsening phase. The complexity of all these schemes is *O (|E|)*.

Random matching The randomized algorithm helps for generating maximal matching more efficiently. A major assumption is vertices are visited in random order including the following steps.

- Consider a graph G (V, E) where a vertex 'u 'has not been matched yet for any of its unmatched adjacent vertex is selected randomly.
- If such a vertex 'v' exists, we include the edge (u, v) in the matching and mark vertex u and v is being matched.
- If there is no unmatched adjacent vertex v, then vertex 'u', remains unmatched in the random matching.

Heavy edge matching (HEM) This technique is much similar as of maximal matching but uses weights of the graph. The main objective of HEM is to find a maximal matching of the graph in addition to minimization of edge cut. In this technique, the vertices are selected in random order. Instead of randomly matching a vertex u with one of its adjacent unmatched vertices, match u with the vertex v such that the weight of edge *(u, v)* is the maximum.

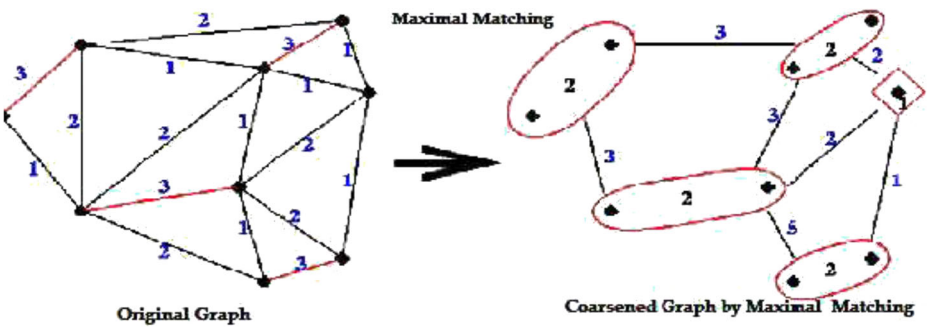

Fig. 12 Graph Coarsening by Maximal Matching [120]

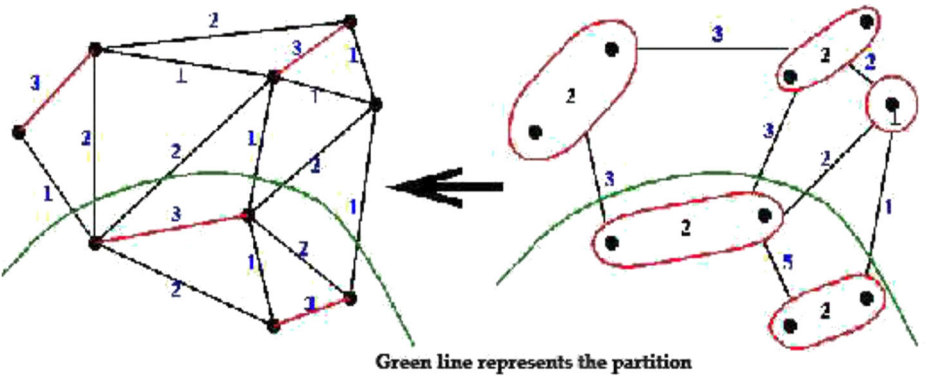

Fig. 13 Converting coarse partition into fine partition [120]

Graph Partitioning A two-way partition P_m of the graph $Gm = (V_m, E_m)$ is computed which partitions $V_m$into two parts, each containing half the vertices of G_0. There are various algorithms available for the graph partitioning; some of them are spectral bisection [30, 41, 97], geometric bisection [32], and combinational methods [38, 104, 109]. Karpys and Kumar developed three algorithms for partitioning of coarser graph. These are spectral bisection [94], graph-growing heuristic (GGP), which randomly selects a vertex v and grows in a breadth-first manner until half of the vertex weight has been integrated. Another is second graph-growing heuristic (GGGP), which also randomly select a vertex v and includes only those vertices whose inclusion leads to a smaller increase in edge-cut. As these both techniques reside on the selection of vertex *v*, different vertex *v* are selected, partitions are computed starting from selected vertices and best is used as initial partition [94].

Graph Uncoarsening and Refinement In uncoarsening phase, the partition of the coarsen graph (P_m) is projected back to the original graph, G_0 by going through intermediate partitions $P_{m-1}, P_{m-2} \ldots P_1, P_0$. While projecting back, it maintains sum of vertex weight in each set. Refinement algorithm selects two sub sets of vertices, one from each part such that when swapped resulting partition has smaller edge cut. If X and Y are two parts of bisection, a refinement algorithm selects $X' \varepsilon X$ and $Y' \varepsilon Y$ such that $X \backslash X' \cup Y'$ and $Y \backslash Y' \cup X'$.

Based on the above-mentioned portioning algorithms, various researchers worked on these algorithms. To summarize few amongst them are -.

Chao – Wei Ou and Sanjay Ranka have suggested parallel incremental graph partitioning using linear programming [29]. It is used to execute several scientific and engineering applications parallel. It requires partitioning of data among processors to balance the computational load on each node with minimum communication overhead. There are many algorithms like geometric, structural, spectral& refinement algorithms are proposed for achieving parallel graph partitioning.

Researcher Stephen Barnard and et.al has suggested Fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems which is used for the partitioning of the graph. It finds application for dynamic graph too. A dynamic graph is a graph, which changes over time that is a small number of nodes or edges may be added or deleted at any given instant. The drawback of the method is initial partition is to be calculated using a linear programming based bisection method [166]. The Proposed approach focuses on uniform partitions creation with no loss of information.

Researcher Inderjit S. Dhillon and et al. has described an equivalence between the objective functions and high-quality multilevel algorithm that optimizes various weighted graph clustering objectives such as ratio cut, normalized cut and ratio association criteria [86].

MahmudurRahman and et al. discussed an efficient graphlet counting method for large graph analysis. It computes the cost for obtaining the frequency of each graphlet in the network [148]. Also the local topological structure is considered for the computation of the Graph Frequency Distribution (GFD).

Vladimir Batagelj and et al. described (X, Y) - clustering and Hybrid visualizations technique for visual analysis of large graphs. In (X, Y) clustering the two important properties like intra cluster and inter cluster were used for topological graph [13]. By using hybrid visualizations clusters were able to explore the clusters without losing their mental map.

Jose F. Rodrigues and et al. discussed large graph analysis in the GMine System for the clutter reduction in the graph which is based on the graph representation as hierarchies of partition by using concept of super graph and sub graph. And graph summarization is performed using Center Piece summarization. Their main contribution is for the large graph investigation in terms of locally and globally [89].

Liu and et al. provides social network analysis, co-authorship networks and their combination [84]. Mainly contributed to computing page rank, authors' Rank and some coefficients of author.

Han and et al. Performed analysis of DBLP dataset to find supportiveness of author [138]. Value of supportiveness is based on co authorship ties in non – symmetric ways.

ZdenekHorak et.al. has developed FORCOA.net as an interactive tool for exploring the significance of authorship network in DBLP data [92]. This tool mainly focuses on analysis and visualization of the co-authorship relationship based on their joint publication and intensity of author. The analysis is performed using forgetting function to hold information relevant to selected date.

6 Graph visualization

It is a representation of interconnected nodes arranged in space and its representation in a particular structure so that user can understand it easily. The size of the graph is too large, leads to a major challenge inefficient graph visualization. It can cause various problem named as algorithmic complexity, display clutter, readability, and navigation -.

i. Algorithmic Complexity: As the graph size is too large, algorithms take more time for its processing. Graph algorithm can visualize algorithms are NP Complete or NP hard. Graph size is vital to algorithms in some cases because a lot of useful graph algorithms are either NP-complete or NP-hard. Therefore, some layout algorithms can be totally unusable when facing graphs of a large size. Algorithms require long processing time and make it hard to interact in real-time.
ii. Display Clutter: Large Graph contains thousands of vertices or more. When the size of the data grows, graph becomes visually cluttered and confusing. It becomes difficult for user to discriminate between nodes and edges.

iii. Readability: Human perception able to visualize only small for graphs including firstly finding the node and secondly finding links between nodes.
iv. Navigation: Navigating large information spaces, such as graphs with thousands of nodes suffer from the problem of viewing a large space on a small display.

6.1 Graph layouts

Graph layout is a representation of a graph containing trees and more general networks. Figure 14 shows Graph Layouts. It is mainly composed of Node Link Layout, Space Division Layout, Space Nested Layout, 3D Layout and Matrix Layout.

6.1.1 Node link layout

It shows a relation among the set of nodes and the set of edges. A Position of the node is computed and nodes connected by drawing curve. Generally accepted rules include evenly distribution of nodes and edges, avoidance of edge crossing, display of isomorphic sub-structure and minimization of edge bends. It is further classified into three types such as Tree layout, Tree+ link layout, and Spring Layout.

Tree Layout: It shows parent child relationship to indicate a link between nodes as shown in Fig. 15a [77]. It suffers from a major problem of inefficient use of screen space. It wastes the root side of the tree and severely clutters the opposite side.

This problem can be avoided by another node-link layout called radial layout as shown in Fig. 15b. It recursively positions children of a sub-tree into a circular wedge shape according to their depths in the tree. Generally, radial views, including its variations share common characteristics. The focus node is always placed at the center of the layout, and the other nodes radiate outward on separated circles.

Balloon layout is shown in Fig. 15c is similar to the radial layout. Balloon layouts are formed where siblings of sub-trees are placed in circles around their father node.

□ Tree+ Link Layout: Tree Plus enables users to interactively explore a graph by starting at a node and then incrementally expanding and exploring the graph. For example, Web Map [71] gives a visualization of the user's web browser history. If a webpage is visited for the first time, a new node then is added and connected with its predecessor as part of the underlying

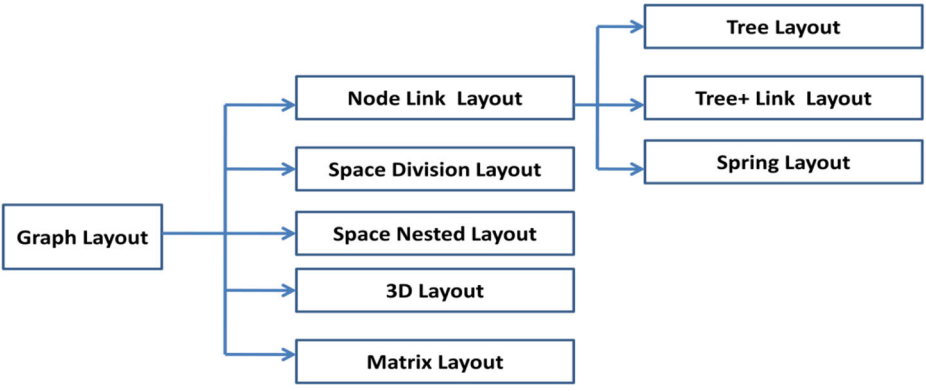

Fig. 14 Graph Layouts

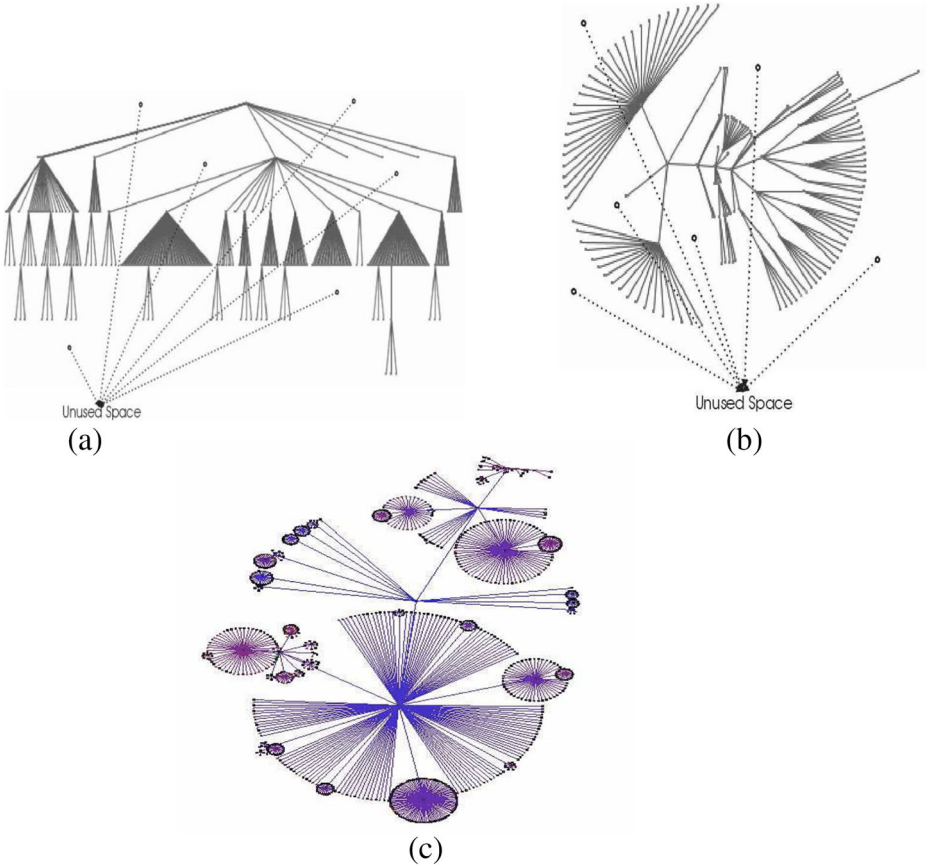

Fig. 15 Node-link layout (**a**) Classical hierarchical view for a moderate largeTree [31] (**b**) Radial view [31]. (**c**) Balloon view [71]

tree. However, if the webpage is visited before, a cross-link is then created to connect the node with its predecessor as shown in Fig. 16.

Spring Layout: It is one of the techniques for graph layout and also known as Force-Directed layout, modeled as physical systems of rings or springs [130].

ii. Space Division Layout: In space division layout, the parent-child relationship is represented by the attachment of the child node(s) to the parent node. Since the parent-child and sibling relationships are both expressed by adjacency, the layout must have a clear orientation to differentiate these two relationships as shown in Fig. 17.

6.1.2 Space nested layout

It uses nested way for drawing hierarchical structure. A rectangle is subdivided into smaller rectangles horizontally or vertically. An outer large rectangle represents parent, while smaller rectangle represents one of its children. Size of rectangle is in proportion with attributes of node. It is referred as Tree maps, as it is most compact display among three layouts [151] (Fig. 18).

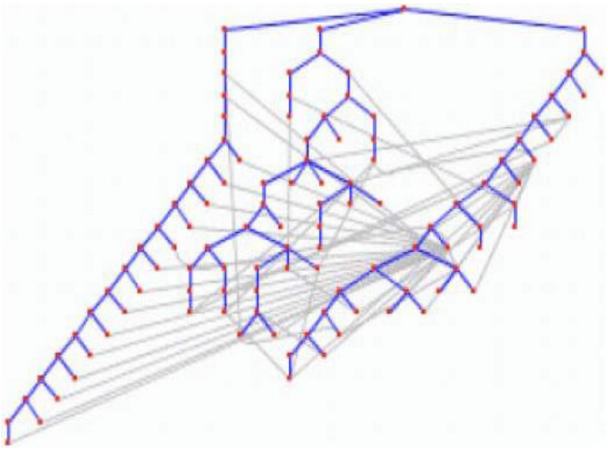

Fig. 16 Tree + Link Layout [47]

6.1.3 3D layout

Many 2 D layouts are extended to 3 D layout. 3D layouts provide an extra dimension in terms of more space for displaying large structures [159]. Cone tree structures as shown in Fig. 19 is one of the examples of the 3D layout. The parent-child relationship is indicated by a cone in which parents are placed at the apex of the cone whereas children are placed at the base of the cone.

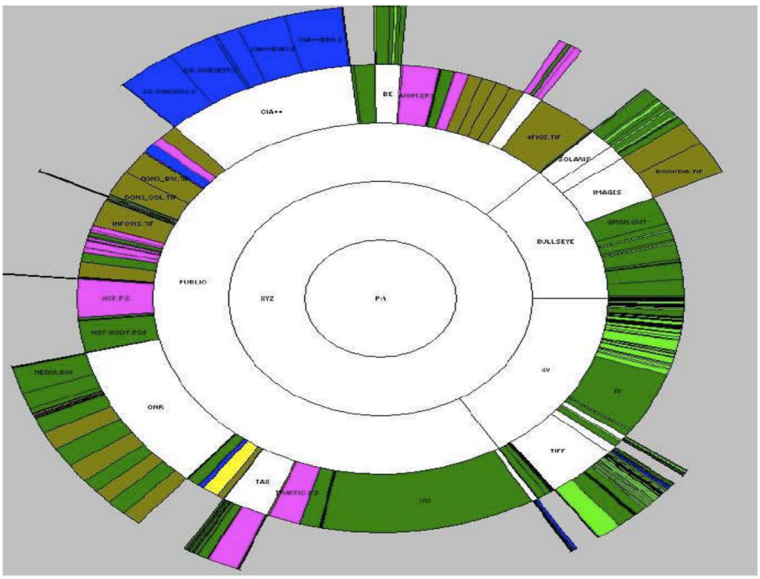

Fig. 17 Sun Burst Visualization of a file directory

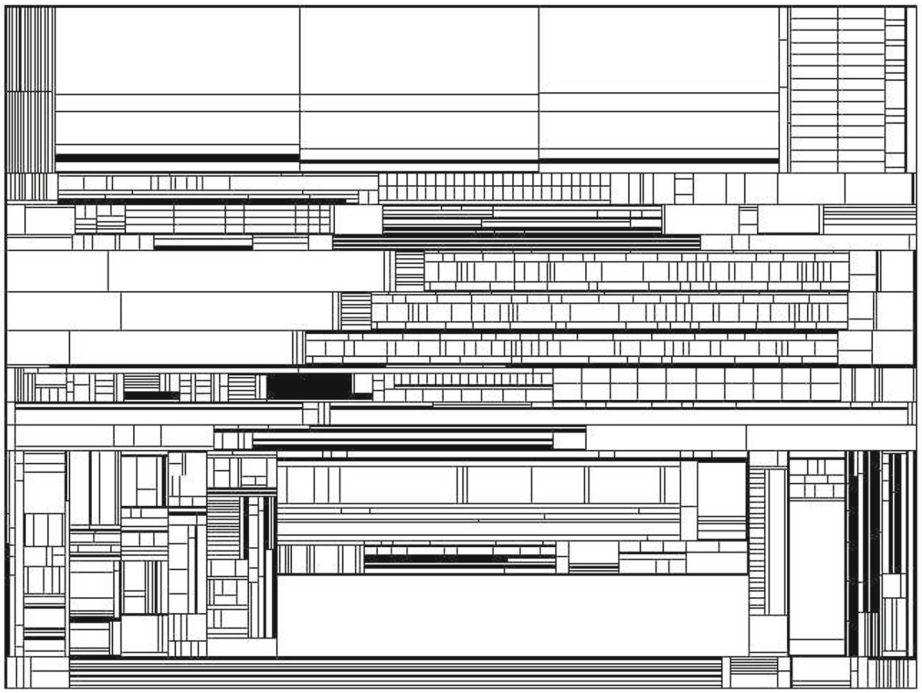

Fig. 18 Tree maps views: Tree maps view of a file directory [177]

6.1.4 Matrix layout

Another approach for graph visualization is the matrix layout. Graphs can be presented by their connectivity matrices as shown in Fig. 2.19. Each row and each column corresponds to a node. If it contains an edge from i to *j*, then it is represented with the interaction of *(i, j)* (Fig. 20).

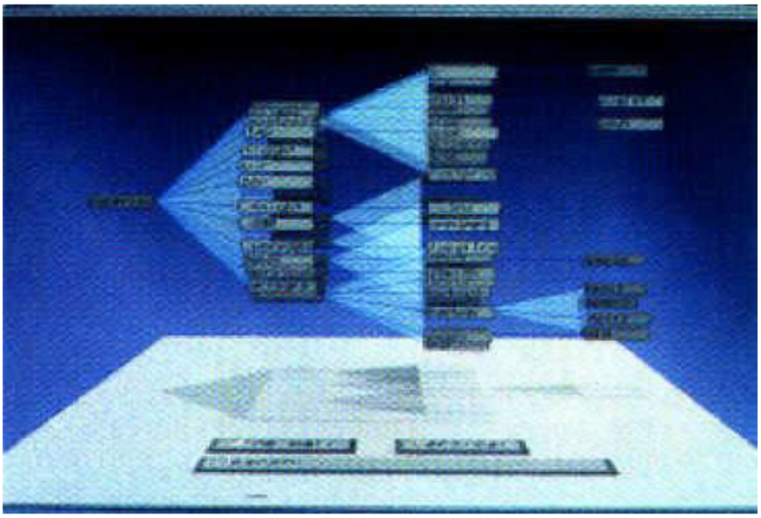

Fig. 19 Cone Tree Layout

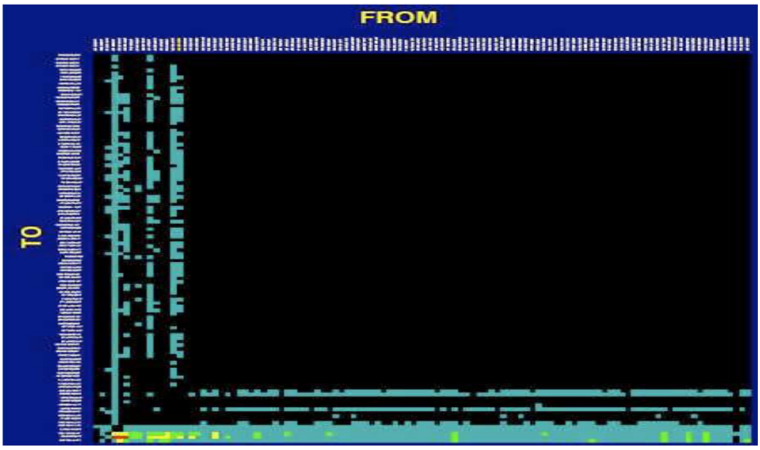

Fig. 20 Matrix Views [167]

6.2 Graph visualization techniques

The large graph problems are cannot be solved using static layout techniques. For the computerized information visualization, the most popular schemes are interaction and navigation. For solving these problems many visualization techniques has been developed. Graph Visualization Techniques are mainly classified into two types namely visual clutter reduction and interaction and navigation as shown in Fig. 21.

6.2.1 Visual clutter reduction

Visual clutter is one of the problems of a large graph. A layout should have minimum visual clutter. The different ways used for it are edge displacement, node clustering, and sampling [47].

Edge Displacement The best method for reducing the edge clutter is to minimize edge crossing. To achieve it, the best solution is to draw edges like splines and polylines. If geographical positions of the graph are fixed, then edge drawing becomes easier. To handle

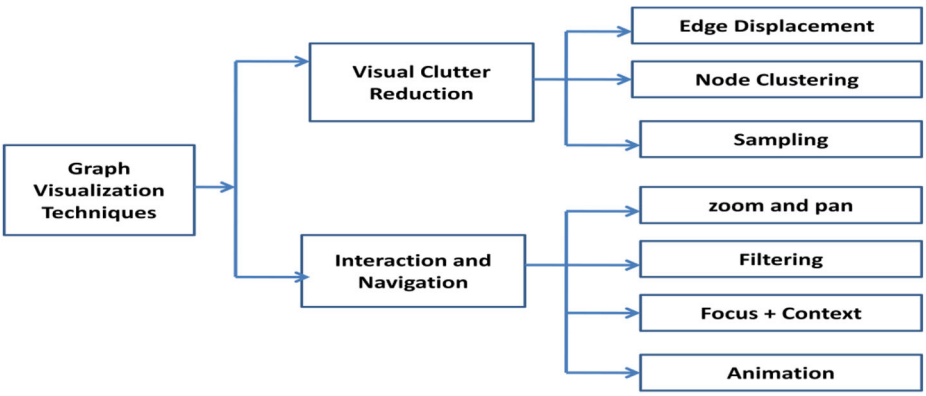

Fig. 21 Graph Visualization Techniques

large graph and find its optimal solution is a very time-consuming task. Hence, if a layered graph is constructed without edge crossing, then it leads to more zig-zag lines, which is not valid regarding the visualization of information.

Some methods have been proposed by the researcher to reduce edges. One of the methods is confluent drawing [49, 79]. In which, the lines are drawn as curves for smooth intersecting arcs and it leads to reduce edge crossing. Two nodes are connected if and only if there is a smooth curve is present without ant sharp turns.

In the confluent drawing layout, the edge-clustering approach is used, which focuses on the reduction of edge covering area instead of edge crossing. With the edge merging, freer space is available which causes visual clutter reduction. The most important feature of edge clustering is that it provides a simple and clear picture of the whole graph. Phan et al. propose an algorithm to generate flow map layouts. A set of edges which shares common endpoints are identified. And those are connected like a free style binary tree.

As shown in Fig. 21, in hierarchical structure formation, common end points are computed and treated as tree roots, then leaf positions are identified & leafs are preserved. The line widths are proportional to edge weights. Flow maps are provided which helps for reduction in visual clutter. It works well for small graph visualization but fails for overlapped flow maps, which generate difficult patters. This drawback can be further improved by another clustering approach known as edge bundles. As shown in Fig. 22, bundle width naturally indicates links connectivity with different parts of hierarchical structure (Fig. 23).

Wong et al. presented edge clutter removal strategy named as edgelens [190]. It is depending upon user's request. In this strategy, node positions are fixed, and edgelens can effectively curve graph edges away from their focus. This structure is shown in Fig. 24 [80].

Node Clustering n general, clustering means a grouping of nodes. It has many applications in various domains such as cluster analysis, grouping, classification, and pattern recognition. For specific context like visual clutter reduction, clustering refers to divide the whole graph into sub-graphs. Representing those sub-graphs as a single node, a grouping of similar elements and free the space. This works well to reduce visualization clutter. Mainly there are two types of node clustering as content-based and structure-based. Mainly there are two types of node clustering as content-based and structure-based.

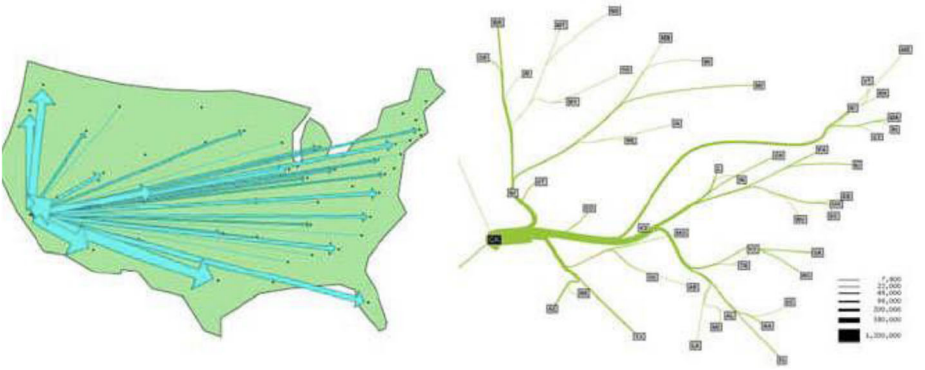

Fig. 22 Confluent drawing of layered drawings [79]

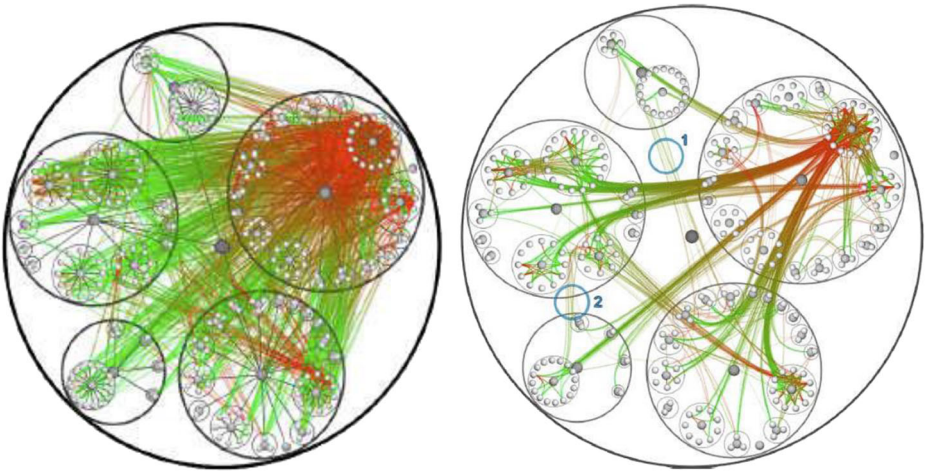

Fig. 23 Hierarchical edge bundles

1. Content-Based: This is application relevant approach suitable for specific problems and not a generalized one. It is based on semantic data related to graph elements. It produces meaningful clustering results and explains a method for aggregation of vertices depending on attributes [130, 151, 177]. It computes attribute values like pivot TABLEs in spreadsheet calculators.
2. Structure-based: This is a generalized approach of clustering and hence more preferred than the content-based approach. In this approach, clusters are generally formed based on graph components that have more intra- link connections than the outside elements. In this type of design various heuristics such as connectivity, cluster size, geometric proximity and statistical variation are proposed [131, 154, 167]. Structure-based approach is classified into three types as graph-theoretical, singe pass and iterative algorithms.
3. Graph Theoretical: In this technique, the similarity between individual nodes is computed. Furthermore, the similarity matrix is formed. Closely related node forms the cluster, where

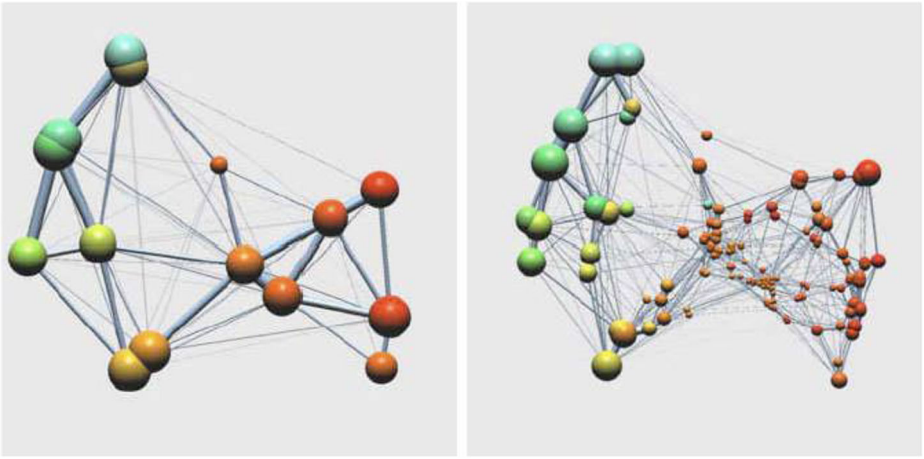

Fig. 24 Different clustering levels of the same graph [95]

each cluster indicates a connected graph. It includes various techniques such as spectral bisection [66], spectral quadra section [93], and octa section [12].

4. Single Pass: In this technique, the individual data seeds are known as cluster seeds. The nodes are added one by one until the big cluster is formed. In this way, clusters are formed by growing individual cluster nodes. [26, 98].
5. Iterative algorithms: It works in iteration and forms a hierarchical structure by considering single pass clustering as a starting point. It merges smaller clusters into one to form a larger one [36, 78, 95, 122]. It involves three more steps such as coarsening, partitioning, projecting and refining of a graph.

But the quality of the graph is dependent on its application domain. Some general characteristic of good clustering algorithm for human perception as-.

- Balanced cluster: At each hierarchy level, each cluster size should be same and distribution of nodes must be as even as possible.
- Small cluster depth: In recursive decomposition, the number of layers should be small.
- Convex cluster drawing: Each cluster should fit in a simple convex region.
- Balanced aspect ratio: Cluster regions should not be too thin. It should be big enough.
- Efficiency: Cluster computation should not take a long time.
- Symmetry: Display balance should be maximized.

Well-arranged clusters can provide clear visualization to the user. Some algorithms provide two or three-dimension representation of the graph is discussed in [43, 46, 55, 179] as shown in Figs. 25.

Sampling It is a new method for visual clutter reduction based on the shape of the degree distribution of a graph [105]. It selects nodes randomly. It is an unpredicted table approach as it provides different sapling results to different users. Therefore how to define a good sampling strategy is a challenging problem. Existing sampling algorithms are based on histogram difference measure and nearest neighbour measure. But Lescovec and Faloutsus proposed two different goals of sampling such as back in time goal and scale down goal [111]. In back in time goal approach the, sampled graph should look like an original graph where as in scale down goal, sampling should have as much as properties of the original graph.

Focus +context Focus +context technique is mainly focused to overcome e drawback of zooming of losing the context. Consider the situation of zooming into a picture, it would be zoomed in a particular area of the picture, more focus on the zoomed area without having an

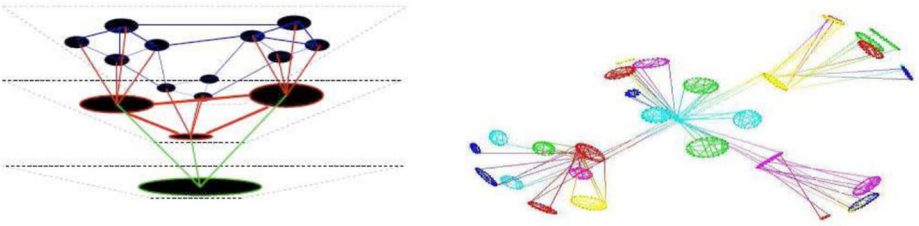

Fig. 25 Clustered graph layouts: (**a**) layered Layout [120] (**b**) 3D Layout [172]

idea about the surrounding area in the picture. Where Focus+ Context provides the ability to see the focused object in a detailed view and overview of surrounding information [163]. Actually, this technique does not replace zoom and pan, but complements them. Majority of visualization system implements both techniques together**.** Fish-eye is one of the most popular focus + context techniques [163].

Animation It is one of the unique visualization techniques as it provides data exploration by adding time as an important factor. Animation helps users to understand the concept without thinking more [115]. Animated transition provides better view of data relations.

6.2.2 Interaction and navigation

Navigation and interaction are essential in computerized information visualization. It can help users to reveal detailed structures in large graphs. Yi et al. proposed a summarization of popular interaction techniques [154]. Based on the purposes, they are categorized into seven 699 groups:

- Select: It helps users to highlight certain focus targets, or request computer to process some specific items.
- Explore: It is used to change the current viewpoint to another part of the data in the same layout representation, such as panning and rotating.
- Reconfiguration: It is used to switch between different layouts with the same representation scheme, such as replacing nodes in graphs and reordering data items based on different criteria.
- Encoding: It is used to switch different representation schemes, such as changing the layout from node-link representation to tree maps representation.
- Abstract/Elaborate: It adjusts the level of abstraction of a data representation to give users different insights into the data, such as zooming and clustering.
- Filter: It reduces the amount of data being displayed and makes the remaining items more visible based on users' requested.
- Connect: connection between different nodes/ items is highlighted.

For large graphs, three of them are especially helpful: explore, abstract/elaborate, and filter.

Remaining strategies like smooth panning and zooming, focus + context and animation to improve the quality of interaction.

7 Assessment of graph mining algorithm

The graph mining algorithms are analyzed with respect to their application areas. The details are discussed in Table 5.

8 Datasets available

Table 6 shows the graph databases with graph examples and features.

Table 5 Application Areas With Performace Measrure Parameter

Application Areas	Parameter	Reported In
VLSI Design	Number of Edge Cuts	[62, 63, 77, 78, 86, 95, 169, 178]
Image Processing	Precision, Recall	[8, 82, 107, 120, 189]
DBLP Processing	Classification	[4, 24, 48, 80, 81, 89, 140, 159, 168, 176]
Medical Fields	Graph Sub Structure	[30, 63]
Graph Clustering	Patterns of Graph	[9, 13, 21, 30, 38, 47, 71, 89, 91, 101, 102, 104, 109, 113, 119, 122, 131, 134, 154, 157, 158, 194]

8.1 Relational databases

The mainstream graph databases provide an object model for nodes and relationships. These graph databases focus on RDF triples, linked data or relationships for storage. These databases often use direct memory links to adjacent nodes rather than requiring joins or keys lookups. The.

□ DEX (2007).

It is a very large graph dataset is written in C++. It supports various operations such as and keyword search through JAVA API. It finds application areas as link analysis, social network analysis, and pattern recognition.

□ Hyper GraphDB (2010): [3] is an open-source database. It provides support for hypergraphs. It is useful in the modeling of graph data for artificial intelligence, bio-

Table 6 Datasets

Name of Dataset	Paper	Example	Features
Relational Datasets	[3]	DEX	link analysis, social network analysis, pattern recognition
	[7]	Hyper GraphDB (2010)	artificial intelligence, bio-informatics, and other knowledge representations
Distributed Graph Databases	[83]	Horton (2010)	allows for better ad-hoc querying
	[83]	In_niteGraph (2010)	competent traversal of graphs across distributed data stores
Key-Value Graph Databases	[173]	VertexDB (2009)	Focuses on a vertex of a graph and supports for automatic garbage collection.
	[33]	CloudGraph (2010)	It takes benefit of key/value pairs to store data both in- memory and on-disk.
Document Graph Databases	[136]	Orientdb(2009)	provide quick data retrieval for structured data
SQL Graph Databases	[57]	Filament(2010)	Allows SQL querying through JDBC with navigational queries for querying the graph data.
	[68]	G-Store(2010)	Storage and retrieval of a graph data from the database.
Map / Reduce Graph Databases	[141]	Phoebus(2010)	building on top of Hadoop in order to beneath from the Map/Reduce framework
	[65]	Giraph(2011)	If the application coordinator has a fault, one of the available nodes will automatically become the new coordinator

informatics, and other knowledge representations. It is used in support of hyper- graph for querying with a Java API.

8.2 Distributed graph databases

This type of graph is used for the distribution of large graphs across a framework. A large graph is partitioned into sub-graphs. The relationships are indicated by connections. The common distributed graph databases are-.

□ Horton (2010): This graph-processing framework is created by Microsoft. This facilitates better ad-hoc querying in comparison to map/reduce systems [83].
□ Innite Graph (2010): This type of graph is useful in distributed systems. It provides a large graph analysis. It is useful in distributed data stores for efficient traversal of graphs.

8.3 Key-value graph databases

This type of graph database is used in an object-related model of the graph. The important characteristic of it is having more horizontal scalability. The examples of key-value graph databases are as follows.

□ VertexDB (2009) [173]: This graph mainly based on a vertex graph. It supports for automatic garbage collection.
□ CloudGraph (2010) [33]: This graph database is written in c#. The important characteristic is it is able to store data in both memory and on disk.

8Document Graph Databases.
Like key-value stores, document-based graph databases introduce a higher level of data complexity for a given node. Some of the document graph databases are as follows.

□ Orientdb(2009) [136] It is a high-performance document-graph database. A novel distributed hash TABLE algorithm in order to get greater parallelism. Document stores, much like key-value stores provide quick data retrieval for structured data.

8.4 SQL graph databases

It provides support for the structured query language. The examples of this graph databases are as follows.

□ Filament (2010) [57]: It is a graph database used for querying through JDBC for querying graph data.
□ G-Store (2010) [68]: This graph database is related only with the retrieval of graph data from the database and not concern about data storage.

8.5 Map / reduce graph databases

Maps reduce approach is used to process large graphs with achieving parallelism. More than one machine performs graph partitioning to provide output as a small-sized graph on each machine. Examples of map-reduce graphs are as follows-.

- Phoebus (2010) [141]: It is implemented with the help of Pregel by using map reduce framework with the help of Hadoop platform.
- Giraph (2011) [65]: It is also implemented with the help of Pregel by addition of fault tolerance feature. In this dataset, if any node fault occurs, then one of the nodes from the available node becomes new coordinator and execution continues.

8.6 Social network analysis

Database systems and Logic Programming stood for DBLP. As a backronym, it has been taken to stand for the Digital Bibliography & Library Project. DBLP has collection more than 3.66 million journal articles, conference papers, and other publications on computer. All important journals and proceedings papers on computer science are tracked (http://dblp.uni-trier.de/xml/).

The file dblp.xml contains all bibliographic records, which make DBLP. It is accompanied by the data type definition file dblp.dtd. You need this auxiliary file to read the XML file with a standard parser [28]. dblp.xml has a simple layout: record 1 to record n.

These tags correspond to the entry types used in BibTEX [39]. DBLP records may be understood as "BibTEX records in XML. Sample DBLP record is shown below.

```
<article key="journals/jmm2/PatilKBK10" mdate="2017-05-26">
<author>Varsha H. Patil</author>
<author>Gajanan K. Kharate</author>
<author>Dattatraya S. Bormane</author>
<author>Snehal M. Kamlapur</author>
<title> Super Resolution for Fast Transfer of Graphics over Internet.</title>
<pages> 71-78</pages>
<year> 2010</year>
<volume> 5</volume>
<journal> Journal of Multimedia</journal>
<number> 1</number>
<ee>https://doi.org/10.4304/jmm.5.1.71-78</ee>
<url>db/journals/jmm2/jmm5.html#PatilKBK10</url>
</article>
```

Various attributes of Record are-.

This record describes an article from CACM. The enclosing article element has two attributes.

- key: the key is a unique key of the record. It shows a UNIX file system with slash separation. The subtrees in the key namespace are for papers published in journals, transactions, and magazines. The second part of DBLP depicts conference series or periodicals. Last part of the key contains a sequence of alphanumeric characters with id's formed from authors name and year of publication.
- □ Mdate.

Mdate is the of last modification of the record. The format of the date is YYYY-MM-DD. It provides the facility of loading recent additions into an application. It contains old versions of records.

□ Title.

This is one of the important element has to exist in every DBLP publication record. It has sub-elements for subscripts, sup elements for superscripts, i elements for italics, and tt for typewriter text style.

□ Pages.

It shows the length of the paper. Preferred style for page numbers is "from-to". For a single page paper, page number without a hyphen is written. For articles in magazines, comma-separated list of page numbers of page ranges are used.

□ Years.

The year element is a four-digit number interpreted according to the Gregorian calendar. For journal articles, it is assumed that the date of publication of the issue is definite. For conference proceedings, the specification of year becomes tricky because sometimes proceedings are not published in the same year in which conference held. Hence year in which conference is held considered as year in a record. For journal articles, the volume and number field are used to specify the issue in which paper appeared.

□ URL and ee.

DBLP record contains two URLs under this field. As URLs are of two types, local and global. Global URL is standard internet URL starts with protocol specification of the form "letter + : (http:, ftp:, ….)". Locsl URLs does not start with a protocol name. ee indicates the position of the electronic edition. ee contains the required link information of ACM and IEEE papers. Usually, the ee fields are "global URLs".

9 Conclusions

A significant development in Graph routing task, graph analysis, and visualization has been seen in last few decades, literature is available in the form of journals, transaction papers, patents, reviews, and surveys. Most of the research papers have been referred for knowing the status of research in the domain and identifying the research gap. This survey paper reviews

most of the papers published on graph mining (up to 2020) and proposes a broad taxonomy of graph mining. Detailed techniques of graph mining along with their pros and cons are focused. Furthermore, it highlights the most common ways of dealing with major issues like graph partitioning, graph classification and graph visualization. It provides most commonly used datasets for different applications. It has been observed that the process of graph partitioning is the most important and challenging one. The major goals of partitioning are to speed up the design process, should be independently designed, original system functionality remains intact, and simplify routine tasks with objectives to minimize interactions between blocks. Its accuracy directly impacts the quality of partitioning and loss of information in the form of connectivity through edges.

References

1. Abello J, van Ham F, Krishnan N (2006) Ask-graph view: a large scale graph visualization system. IEEE Trans Visualization and Comput Graph 12(5):669–676
2. Abnar A, Takaffoli M, Rabbany R, Zaiane OR (2014) SSRM: structural social role mining for dynamic social networks advances in social networks analysis and mining (ASONAM). IEEE/ACM international conference 289-296
3. Allegrograph. (n.d.) http://www.franz.com/agraph/allegrograph/
4. Alwahaishi S, Martinovic J, Snasel V, Kudelka M (2011) Analysis of the DBLP Publication Classification Using Concept Lattices. ISBN: 978–80–248-2391. 132–139
5. Andreev K, Racke H (2006) Balanced graph partitioning. Theory Computation System 39(6):929–939
6. Auber D, Chiricota Y, Jourdan F, Melancon G (2003) Multiscale visualization of Small world networks. Proc IEEE Ninth Conf Information Visualization (InfoVis):75–78
7. AUTHOR (n.d.) http://www.hypergraphdb.org/
8. Ayati M, Erten S, Mark Chance R, Koyutrk M (2015) MOBAS: identification of disease-associated protein sub networks using modularity-based scoring. EURASIP J Bioinf Systems Bio 2015:7
9. Bansal N, Blum A, Chawla S (2004) Correlation clustering. Mach Learn 56(1–3):89–113
10. Bapodra M (2009) Chemical Reaction Rate Analysis using Graph Transformations. CO3120 Computer Science Project, Final Report, submitted to the University of Leicester in Partial Fulfillment for the degree of Bachelor of Science, Department of Computer science, University of Leicester 1–89
11. Barabasi AL, Albert R (1999) Emergence of scaling in random networks. Science 286:509–512
12. Barnad ST, Small I (1990) Animation at the Interface: the art of human computer Interface design 251-267
13. Batagelj V, Brandenburg FJ, Didimo W (2011) Visual analysis of large graphs using (X, Y)- clustering and hybrid visualizations. IEEE Trans Knowl Data Eng 17(11):1587–1598
14. Benkö G, Flamm C, Stadler PF (2003) A graph-based toy model of chemistry. J Chem Inf Comput Sci 43: 1085–1093
15. Berlingerio M (2009) Graph and network data: mining the temporal dimension. IMT Institute for Advanced Studies, Lucca
16. Beucher S, Lantuéjoul C (1979) Use of watersheds in contour detection. in International Workshop on Image Processing. Real-Time Edge and Motion Detection/Estimation, Rennes
17. Bolz J, Farmer I, Grinspun E, Schrooder P (2003) Sparse matrix solvers on the GPU: conjugate gradients and multigrid. ACM Trans Graphics (TOG) 22(3):917–924
18. Borgatti SP, Mehra A, Brass DJ, Labianca G (2009) Network analysis in the social sciences. Science 323: 892–895
19. Brath R, Jonker D (n.d.) Graph Analysis and Visualization: Discovering Business Opportunity in Linked Data. John Wiley & Sons, ISBN-10: 1118845846, ISBN-13: 978–1118845844
20. Bui T, Jones C (1993) A heuristic for reducing fill in sparse matrix factorization. In6th SIAM Conf. Parallel processing for scientific Computing 445–452
21. Bulò SR, Pelillo M (2013) A game-theoretic approach to hyper graph clustering. IEEE Trans Pattern Anal Mach Intell 35(6):1312–1327
22. Callut J, Fraisse K, Saerens M, Dupont P (2008) Semi-supervised classification from discriminative random walks. Lecture notes in artificial intelligence no. 5211, springer 162-177

23. Callut J, Françoisse K, Saerens M, Dupont P (2008) Semi-supervised Classification from Discriminative Random Walks. Lecture Notes in Artificial Intelligence No. 5211, Springer 162–177
24. Carpendale GS (2003) Edge Lens: An Interactive Method For Managing Edge Congestion In Graphs. Information Visualization INFOVIS 2003. IEEE Symposium 51–58
25. Chao-Wei O, Ranka S (1999) Parallel incremental graph partitioning. IEEE Transactions on Parallel and Distributed Systems 8(8):884–896
26. Chartrand G, Oellermann OR (1993) Applied and algorithmic graph theory. McGraw-Hill, New York
27. Charu Aggarwal C (n.d.) Managing and Mining Graph Data. Springer 2010 edition, ISBN-10: 1441960449, ISBN-13: 978–1441960443
28. Charu Aggarwal C, Wang H, (n.d.) Managing and Mining Graph Data (Advances in Database Systems). Springer 2010 edition, ISBN-10: 1441960449, ISBN-13: 978–144196044
29. Chaw Wei O, Ranka S (1997) Parallel Incremental Graph Partitioning. IEEE transactions on Parallel and Distributed Systems 8:884–896
30. Chen Y, Fonseca F (2004) A bipartite graph co-clustering approach to ontology mapping
31. Cheng C-K, Wei Y-CA (1991) An improved two-way partitioning algorithm with STABLE performance. IEEE Transactions on Computer Aided Design 10(12):1502–1511
32. Clique BD (2008) Matrices for Statistical Graph Decomposition and Paramenite Matrices. In D. A. McAllester and P. Myllymaki, editors AUAI Press 26–33
33. Cloudgraph. (n.d.) http://www.cloudgraph.com/
34. Corradini A, Montanari U, Rossi F, Ehrig H, Heckel R, Löwe M (1997) Algebraic Approaches To Graph Transformation. Part I: Basic concepts and double pushout approach, in: G. Rozenberg, editor, Handbook of Graph Grammars and Computing by Graph Transformation, Volume 1: Foundations, World Scientific 163–246
35. David H (1987) Statecharts: a visual formalism for complex systems. Science Computer Programming 8: 231–274
36. Delest M, Bordeaux F, Fedou JM, Antipolis NS, Jean-Marc F, Melancon G, Montpellier F (2006) A Quality Measure for Multi-Level Community Structure. SYNASC 8th International Conference
37. Dey S (n.d.) Graph Theory with Applications. Shroff, 1st Edition, ISBN-10: 9350236796, ISBN-13: 978–9350236796
38. Dhillon I, Guan Y, Kulis B (2005) A Fast Kernel-based Multilevel Algorithm for Graph Clustering. Proceedings of The 11th ACM SIGKDD, Chicago, IL, 21–24
39. Diane Cook J, Lawrence Holder B, (n.d.) Mining Graph Data. Wiley-Blackwell, ISBN-10: 0471731900, ISBN-13: 978–0471731900
40. Dias CR, Ochi LS (2003) Efficient evolutionary algorithms for the clustering problem. L:983-988
41. Dias CR, Ochi LS (2003) Efficient evolutionary algorithms for the clustering problem in directed graphs. Proceedings of the 2003 IEEE congress on evolutionary computation 1:983-988
42. Duchenne O, Bach F, Kweon IS, Ponce J (2011) A tensor-based algorithm for high-order graph matching. IEEE Trans Pattern Anal Mach Intell 33(12):2383–2395
43. Eades P, Feng QW (1996) Multilevel visualization of clustered graphs. Graph drawing. Proc. 4th Int. Symp. GD 101–112
44. Eades P, Feng Q (1997) Multilevel visualization of clustered graphs. Proc. symposium on graph drawing 101-112
45. Eades P, Huang ML (2000) Navigating clustered graphs using force-directed methods. Graph Algorithms Appl 4:157–181
46. Eades P, Feng QW, Lin X (1996) Straight-line drawing algorithms for hierarchical graphs and clustered graphs. Proceedings of the symposium on graph drawing 113–128
47. Ellis G, Dix A (2007) Taxonomy of clutter reduction for information visualization. IEEE Trans Vis Comput Graph 13(6):1216–1223
48. Elmacioglu E, Lee D (n.d.) On Six Degrees of Separation in DBLP-DB and More
49. Eppstein D, Goodrich MT, Meng JY (2007) Confluent Layered Drawings AlgorithMica 47(4):439–452
50. Estrada E (2013) Chemical Graph Theory. https://www.researchgate.net/publication/258021291, Chapter. 1–25
51. Faloutsos M, Faloutsos P, Faloutsos C (1999) On power-law relationships of the internet topology. Proc. ACM Conf. Applications, technologies, architectures, and protocols for computer Communication 251–262
52. Faloutsos C, McCurley KS, Tomkins A (2004) Fast discovery of connection subgraphs. Proc ACM 10th Int'lConf Knowledge discovery and data mining (SIGKDD) 118-127
53. Fatat GD, Berthold MR (2005) High performance subgraph Mining in Molecular Compounds. HPCC: 866–877

54. Felzenszwalb PF, Huttenlocher DP (2004) Efficient graph-based image Segmentation. Int J Comput Visual 59(2):167–181
55. Feng QW, Cohen RF, Eades P (1995) How to draw a planar clustered graph. Proceedings of the first annual international conference on computing and combinatory, 21–30
56. Fiduccia CM, Mattheyses RM (1982) A linear-time heuristic for improving network partitions. In 19th design automation conference (pp 175–181). IEEE
57. Filament. (n.d.) http://filament.sourceforge.net
58. Finocchi I (2002) Hierarchical decompositions for visualizing large graphs. PhD diss., PhD thesis, Universita degli Studi di Roma "La Sapienza
59. Fortunato S (2010) Community detection in graphs. Phys Rep 486(3–5):75–174
60. Takes FW (2014) Algorithms for analyzing and mining real-world graphs. PhD diss., Leiden University
61. Gao S, Li MM (2014) Research of data graph mining based on telecommunication customers. Appl Mech Mater 443. Trans Tech Publications Ltd
62. Garbers J, Promel HJ, Steger A (1990) Finding clusters in VLSI circuits. In 1990 IEEE International Conference on Computer-Aided Design 1990 Jan 1 (pp 520–521). IEEE Computer Society
63. Garey MR, Johnson DS (1983) Crossing number is NP-complete. SIAM J Alg Discr Meth 4(3):312–316
64. Gentilini R, Piazza C, Policriti A (2003) Computing strongly connected components in a linear number of symbolic steps. InSODA 3:573–582
65. Giraph. (n.d.) https://github.com/apache/giraph
66. Golub GH, Van Loan CF (1996) Matrix computations. John Hopkins University Press Baltimore, Baltimore
67. Gomez-Rodriguez M, Leskovec J, Krause A (2010) Inferring networks of diffusion and influence. Proc. 16th ACM SIGKDD Int'l Conf. Knowledge discovery and data mining 1019-1028
68. G-store. (n.d.) http://g-store.sourceforge.net/
69. Guo P, Wang L, Chen P (2014) A performance modelling and optimization analysis tool for sparse matrix-vector multiplication on GPUs. Parallel and Distributed Systems, IEEE Trans 25(5):1112–1123
70. Guo G, Wang H, Bell D, Bi Y, Greer K (n.d.) KNN Model-Based Approach in Classification. 1–12
71. Hagen L, Kahng A (1992) A New Approach To Effective Circuit Clustering. In: Proceedings of IEEE International Conference on Computer Aided Design 422–427
72. Han J, Yan X, Yu PS (2006) Mining and searching graphs and structures. Proceedings of 12th ACM conference on knowledge discovery and data mining (SIGKDD'2006)
73. Hans Othmer G (1981) A graph-theoretic analysis of chemical reaction networks. Department of Mathematics University of Utah Salt Lake City, Utah 84112:1–36
74. Haq A (2015) Applying graph mining techniques to solve complex software engineering problems. Kent State University
75. Harchaoui Z, Bach F (n.d.) Image Classification with Segmentation Graph Kernels. 1–6
76. Heath MT, Raghavan P (1994) A Cartesian nested dissection algorithm. Technical report UIUCDCS-R-92-1772, Department of Computer Science, University of Illinois, Urbana, IL 61801, 1992. To appear in SIAM journal on matrix analysis and applications
77. Hendrickson B, Leland R (1993) A multilevel algorithm for partitioning graphs. Technical report SAND93-1301, Sandia National Laboratories
78. Hendrickson B, Leland R (1995) A multilevel Algorithmfor partitioning graphs. Proc. Supercomputing
79. Ho J, Hong SH (2006) Drawing clustered graphs in three dimensions
80. Holten D (2006) Visualization of adjacency relations in hierarchical data. IEEE Trans Vis Comput Graph 12(5):741–748
81. Horak Z, Kudelka M, Snasel V, Abraham A, Rezankova H (2011) Forcoa.NET: An Interactive Tool for Exploring the Significance of Authorship networks in dblp data. ieee computer society, IEEE
82. Horak Z, Kudelka M, Snasel V, Abraham A (2011) Forcoa.NET: An Interactive Tool for Exploring the Significance of Authorship Networks in DBLP Data. International Conference on Computational Aspects of Social Networks (CASON) 978–1–4577-1133-6/11/ IEEE, 261–266
83. Horton. (n.d.) http://research.microsoft.com/en-us/projects/ldg/
84. Huang DH, Kahng AB (1995) When clusters meet partitions: new density-based methods for circuit decomposition. In: Proc Eur Conf Des Test 60–64
85. Huang ML, Nguyen QV (2007) A space efficient clustered visualization of large graphs. Proc. fourth Int'l Conf. Image and graphics 920-927
86. Inderjit Dhilon S, Guan Y, Kuli B (2007) Weighted graph cuts without eigenvectors: a multilevel approach. IEEE Trans Pattern Anal Mach Intell 29(11):1944–1957
87. Jalali M, Mustapha N, Sulaiman MN, Mamat A (2010) Expert systems with applications. Elsevier 37: 6201–6212

88. JayantKulkarni S (2017) Graph theory: applications to chemical engineering and chemistry. Galore Int J Appl Sci Human
89. Jose Rodrigues F Jr, Pan J-Y, Agma Traina JM, Traina C Jr, Faloutsos C (2013) Large graph analysis in the Gmine system. IEEE Trans Knowl Data Eng 25:106–119
90. Kang U, Meeder B, Papalexakis EE, Faloutsos C (2014) Heigen: spectral analysis for billion-scale graphs. Knowl Data Eng IEEE Trans 26(2):350–362
91. Kannan R, Vempala S, Vetta A (2004) On Clustering's good, bad and spectral. J ACM 51(3):497–515
92. Kappes JH, Speth M, Andres B, Reinelt G, Schn C (2011) Globally optimal image partitioning by multicuts. In: Proc. energy minim. Methods computation. Vis. Pattern Recognition 31–44
93. Karpys G, Kumar V (1995) Metis: unstructured graph partitioning and sparse matrix ordering system. Version 2.0, University of Minnesota
94. Karypis G, Kumar V (1995) Multilevel graph partitioning schemes. Proc IEEE/ ACM conf Parallel processing 113–122
95. Karypis G, Kumar V (1998) Multilevel algorithms for multi-constraint graph partitioning. Supercomputing 1998. SC98. IEEE/ACM conference 28–28
96. Kashima H, Inokuchi A (2002) Kernels for graph classification. ICDM, Workshop on Active Mining
97. Kazimianec M, Augsten N (2013) Clustering with proximity graphs: exact and efficient algorithms. International Journal of Knowledge based Organizations 3:84–104
98. Kernighan BW, Lin S (1970) An efficient heuristic procedure for partitioning graphs. Bell System Technical Journal 49(2):291–307
99. Ketkar NS, Holder LB, Cook OJ (2009) Empirical comparison of graph classification algorithms. IEEE
100. Khuller S, Saha B (2009) On finding dense subgraphs. In: Proc. automata languages Program 597–608
101. Kim S (2003) Graph theoretic sequence clustering algorithms and their applications to genome comparison. in: J.T.L. Wang, C.H. Wu, P.P. Wang (Eds.), Computational Biology and Genome Informatics, World Scientific Publishing Company 81–116
102. Kim S, Nowozin S, Kohli P, Yoo CD (2011) Higher-order correlation clustering for image Segmentation. In: Proc. Adv. Neural Inf. Process. System 1530–1538
103. Kramer S, Raedt. LD (2001) Feature Construction With Version Spaces For Biochemical Application. In Proc. of the 18th ICML
104. Kraus JM, Palm G, Kestler HA (2007) On the robustness of semi supervised hierarchical graph clustering in functional genomics
105. Krishnamurthy V, Faloutsos M, Chrobak M, Lao L, Cui JH, Percus AG (2005) Reducing large internet topologies for faster simulations. IFIP Networking
106. Kudelka M, ZdenekHorak VS, Abraham A (2010) Social network reduction based on stability. IEEE Computer Society, IEEE
107. Kudelka M, Horak Z, Snasel V, Abraham A (2010) Social Network Reduction Based on Stability. International Conference on Computational Aspects of Social Networks 510–514
108. Laukens K, Naulaerts S, VandenBerghe W (2014) Bioinformatics approaches for the functional interpretation of protein lists: from ontology term enrichment to network analysis. Proteomics 15(5–6):98–196
109. Le TV, Kulikowaski CA, Muchnik IB (2008) Coring method for clustering a graph. In: Proceedings of IEEE
110. Lee J, Cho M, Lee KM (2011) Hyper graph matching via reweighted random walks. In proc. IEEE Conf. Computation visualization pattern recognition 1633–1640
111. Leskovec J, Faloutsos C (2006) Sampling from large graphs. Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining 631–636
112. Ley M (2009) DBLP- some lessons learned. VLDB, August
113. Lin F, Cohen WW (2010) Power iteration clustering. In proc. Int. Conf. Mach. Learn 10:655–662
114. Liu W, Li Z, Sun S, Gupta M, (2021) Design a novel target to improve positioning accuracy of autonomous vehicle navigation system in GPS denied environments IEEE transactions on industrial informatics
115. Lombaert Herve, Yiyong Sun, Leo Grady, Chenyang Xu (2005) A Multilevel Banded Graph Cuts Method for Fast Image Segmentation. Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV, 1550–5499/05 $20.00 © 2005 IEEE'05)
116. Low Y, Bickson D, Gonzalez J, Guestrin C, Kyrola A, Hellerstein JM (2012) Distributed graph lab: a framework for machine learning and data mining in the cloud. Proc VLDB Endowment 5:716–727
117. Lowe DG (2004) Distinctive image features from scale-invariant Keypoints. Int J Comput Visual 60(2): 91–110
118. Ma Y, Li Z, Malekian R, Zheng S, Sotelo MA (2020) A Novel Multimode Hybrid Control Method for Cooperative Driving of an Automated Vehicle Platoon. IEEE Internet Things J 8:5822–58-38

119. Macskassy S, Provost F (2007) Classification in networked data: a toolkit and a univariate case study. J Mach Learn Res 8:935–983
120. Malcolm J, Rathi Y, Tannenbaum A (2007) Graph Cut Segmentation with Nonlinear Shape Priors. IEEE international conference on ICIP 365–368.
121. Malewicz G, Austern MH, Bik AJ, Dehnert JC, Horn I, Leiser N, Czajkowski G (2010) Pregel: a system for large-scale graph processing. In: Proc. ACM SIGMOD Int Conf Manage Data 135–146
122. Mancoridis S, Mitchell BS, Chen Y, Gansner ER (1999) Bunch: a clustering tool for the recovery and maintenance of software system structures. Software maintenance. 1999.(ICSM'99) proceedings. IEEE international conference 50–59
123. Mansurul Bhuiyan A, Mohammad Al Hasan (2013) An iterative map reduce based frequent subgraph mining algorithm. Trans Knowl Data Eng
124. MarziehAyati SE, Mark Chance R, Koyutrk M (2015) MOBAS: identification of disease-associated protein sub networks using modularity-based scoring. J Bioinf Systems Bio
125. Meysman P, Sanchez-Rodrguez A, Fu Q, Marchal K, Engelen K (2013) Expression divergence between Escherichia Coli and Salmonella EntericaSerovarTyphimuriumReflects their lifestyles. Mol Biol Evol 30(6):1302–1314
126. Meysman P, Saeys Y, Sabaghian E, Bittremieux W, Van de Peer Y, Goethals B, Laukens K (2016) Mining the enriched subgraphs for specific vertices in a biological graph. IEEE/ACM, Trans Comput Bio Bioinf
127. Meysman P, YvanSaeys ES, WoutBittremieux YVde P, Goethals B, Laukens K (2016) Mining the enriched subgraphs for specific vertices in a biological graph. IEEE/ACM Transactions On Computational Biology And Bioinformatics
128. Milenkovic T, Przulj N (2008) Uncovering biological network function via Graphlet degree signatures. Cancer Informat 6:257–273
129. Motoda H (2006) What can we do with graph-structured data a data mining perspective. Springer 1-2
130. Mukherjea S, Foley JD, Hudson SE (1995) Visualizing complex hypermedia networks through multiple hierarchical views
131. Newbery FJ (1989) Edge connection: a method for clustering directed graphs. Proceeding of 2nd international workshop on software configuration management 76-85
132. Newman M (2005) A measure of betweenness centrality based on random walks. Soc Networks 27:39–54
133. Newman M (2006) Modularity and community structure in networks. Proc Nat Acad Sci 103(23):8577–8582
134. Ng AY, Jordan MI, Weiss Y (2002) On spectral clustering: analysis and an algorithm. Adv Neural Inf Process System 2:849–856
135. Nour-Omid B, Raefsky A, Lyzenga G (1986) Solving finite element equations on concurrent computers. In: Noor AK (ed) . American Soc. Mech. Eng, pp 291–307
136. Orientdb. (n.d.) http://www.orientdb.org/
137. Ozaki T, Ohkawa T (2008) Mining correlated subgraphs in graph databases. PAKDD 272-283
138. Papadimitriou C, Steiglitz K (1998) Combinatorial optimization: algorithms and complexity. Dover, Mineola
139. Peng J, Xiao Z, Chen C, Yang W (2016) Iterative Sparse Matrix-Vector Multiplication on In-Memory Cluster Computing Accelerated by GPUs for Big Data. IEEE 978–1–5090-4093-3/16/, ©2016
140. Phan D, Xiao L, Yeh R, Hanrahan P, Winograd T (2005) Flow map layout. IEEESymposium on Inf Visual INFOVIS:219–224
141. Phoebus. (n.d.) https://github.com/xslogic/phoebus
142. Ponnusamy R, Mansour N, Choudhary A, Fox GC (1993) Graph contraction and physical optimization methods: a quality-cost trade off for mapping data on parallel computers. In International Conference of Supercomputing
143. Porikli F (2013) Integral Histogram: A Fast Way To Extract Histograms In Cartesian Spaces in Proc IEEE Conf Computation Vis Pattern Recognition 1
144. Pothen A, Horst Simon D, Liou K-P (1990) Partitioning sparse matrices with eigenvectors of graphs. SIAM Journal of Matrix Analysis and Applications 11(3):430–452
145. Pothen A, Simon HD, Liou KP (1990) Partitioning sparse matrices with Eigen- vectors of graphs. SIAM J Matrix Anal 11:430–452
146. Preciado V, Jadbabaie A (2013) Moment-based spectral analysis of large-scale networks using local structural information. IEEE/ACM Trans Netw 21(2):373–382
147. Przulj N (2007) Biological network comparison using Graphlet degree distribution. Bioinformatics 23(2): e177–e183
148. Rahman M, Bhuiyan MA, Al Hassan M (2014) GRAFT: an efficient Graphlet counting method for large graph analysis. IEEE Trans Pattern Anal Mach Intell 26(10):2466–2478

149. Ramon J (n.d.) Graph mining declarative languages and artificial intelligence group. K. U. Leuven, Belgium
150. Rehman SU, Khan AU, Khan AU (n.d.) Graph Mining: A Survey of Graph Mining Techniques. ©2012 IEEE 978–1–4673-2430-4112
151. Risch JS, Rex DB, Dowson ST, Walters TB, May RA, Moon BD (2006) The starlight information visualizations of web content. Int J Hum Comput Stud 53(5):695–714
152. Rodrigues Jr. JF, Tong H, Traina AJM, Faloutsos C, Leskovec J (2006) GMine: A System for Scalable, Interactive Graph Visualization and Mining. Proc. 32nd Int'l Conf. Very Large Data Bases (VLDB) 1195–1198
153. Rossell'O F, Valiente G (2005) Chemical graphs, chemical reaction graphs, and chemical graph transformation. Elsevier, Electronic Notes in Theoretical Computer Science 127:157–166
154. Sablowski R, Frick A (n.d.) Automatic Graph Clustering. Proc. of 4th Symposium on Graph Drawing 395–400
155. Saha B, Hoch A, Khuller S, Raschid L, Zhang XN (2010) Dense subgraphs with restrictions and applications to gene annotation graphs. In proc. res. Computation Mol Biol 456–472
156. Salihoglu S, Widom J (2013) GPS: a graph processing system. In: Proc. 25th Int. Conf. Sci. Statist. Database Manage 22:1:22–12
157. Schaeffer SE (2007) Graph clustering. Computer Science Review 1:27–64
158. Schaeffer SE (2007) Survey graph clustering. Else wire 27–64
159. Schenker A, Last M, Bonke H, Kandel A (2003) Classification of web documents using a graph mode. Proceedings of the Seventh International Conference on Document Analysis and Recognition
160. Seo S, Yoon EJ, Kim J, Jin S, Kim JS, Maeng S (2010) Hama: an efficient matrix computation with the mapreduce framework. In cloud computing technology and science (CloudCom), 2010 IEEE second international conference on. IEEE 721–726
161. Shao B, Wang H, Li Y (2013) Trinity: a distributed graph engine on a memory cloud. In: Proc. ACM SIGMOD Int. Conf. Manage. Data 505–516
162. Shi J, Malik J (1997) Normalized cuts and image Segmentation. In IEEE Conference on Computer Vision and Pattern Recognition 73:1–737
163. Shi J, Malik J (2000) Normalized cuts and image Segmentation. IEEE Trans Pattern Anal Mach Intell 22(8):888–905
164. Shiy J, Malik J (n.d.) Normalized Cuts and Image Segmentation. Supported by (ARO) DAAH04–96-1-0341
165. Snow D, Viola P, Zabih R (2000) Exact voxel occupancy with graph cuts. In IEEE Conference on Computer Vision, and Pattem Recognition 1:345–352
166. Stephan Barnad T, Horst Simon D (1994) Fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. Concurrency: Practice and Experience 6(2):101–117
167. Sugiyama K, Misue K (1991) Visualization of structural information: automatic drawing of compound digraphs. IEEE Trans Syst Man Cybern 21(4):876–892
168. Swell M (2009) Kernel methods
169. Tony Chan F, John Gilbert R, Teng S-H (1994) Geometric spectral partitioning. Technical Report in Preparation
170. van Ham F, van Wijk JJ (2004) Interactive visualization of Small world graphs. Proc. IEEE symposium information visualization (information visualization) 199-206
171. van Ham F, van Wijk JJ, Eindhoven TU (2004) Interactive visualization of Small world graphs. Information visualization. 2004, INFOVIS 2004. IEEE symposium 199–206
172. Veksler O (2008) Star shape prior for graph-cut image Segmentation. Proceeding of the 10th European conference on computer vision
173. Vertexdb. (n.d.) http://www.dekorte.com/projects/opensource/vertexdb/
174. Vitter JS (2001) External memory algorithms and data structures: dealing with massive data. ACM Computing Survey 33(2):209–271
175. Wang L (2006) Comparison for edge detection of Colony image. IJCSNS, Int J Comput Sci Netw Security 6(9A)
176. Wang S, Xiao CL, Liu W (2006) Parallel Enumeration of Custom Instructions Based on Multi depth Graph Partitioning. IEEE Embedded Systems 1–4
177. Wattenberg M (2006) Visual exploration of multivariate graphs. Proceedings of the SGCHI conference on human factors in computing system 811-819
178. Watts DJ, Strogatz SH (1998) Collective dynamics of Small- world' networks. Nature 393:440–442
179. Watts DJ, Worlds S (2003) The dynamics of networks between order and randomness. Princeton Univ. Press

180. Wernicke S, Rasche F (2006) FANMOD: a tool for fast network motif detection. Bioinformatics (Oxford, England) 22(9):115–123
181. Wernicke S, Rasche F (2006) Fanmod: a tool for fast network motif detection. Bioinformatics 22(9):1152–1153
182. Wernicke S, Rasche F (2006) FANMOD: a tool for fast network motif detection Bioinf7ormatics. (Oxford, England) 22(9):115–123
183. Westbrook JD, Fitzgerald PMD, The PDB (n.d.) Format, Mmcif Formats And Other Data Formats
184. Wu X, Wang Y (2008) Interactive For reground/Background Segmentation Based on Graph Cut. Congress on Image and Signal processing 692–696
185. Wu MQY, Faris R, Ma K-L (2015) Visual exploration of academic career paths. 2013, IEEE/ ACM International Conference on Advances in Social Networks Analysis and Mining 779–786
186. Xuan J, Lu J, Zhang G, Luo X (2015) Topic model for graph mining. IEEE Transactions on Cybernetics 2168–2267
187. Yang X, Parthasarathy S, Sadayappan P (2011) Fast sparse matrix-vector multiplication on Gpus: implications for graph mining. Proceedings of the VLDB Endowment 4(4):231–242
188. Stasko J, Yi JS, Kang Y n, Jacko JA (2007) Toward a deeper understanding of the role of interaction in information visualization. IEEE Trans Vis Comput Graph 13(6):1224–1231
189. Yi F, Moon I, Segmentation I (2012) A Survey of Graph-cut Methods. 2012 International Conference on Systems and Informatics (ICSAI 2012), 978–1–4673-0199 /12©2012 IEEE, 1936–1941
190. Yu Z, Xu M, Gao Z (2011) Biomedical image segmentation via constrained graph cuts and presegmentation. International conference of the IEEE on EMBC 5714-5717,
191. Yuri Y Marie-Pierre B Jolly Interactive (2001) Graph Cuts for Optimal Boundary &Region Segmentation of Objects in N-D Images. 0–7695–1143-0/01
192. Zass R, Shashua A (2008) Probabilistic graph and hyper graph matching. In proc. IEEE Conf. Comput. Vis. Pattern Recog. 1–8
193. Zhao P, Yu X (2007) Mining Closed Frequent Free Trees in Graph Databases. Proceeding of Database Systems for Advance Application 91–102
194. Zhou D, Huang J, Schölkopf B (2006) Learning with hyper graphs: Clustering, classification, and embedding. In: Proc. Adv. Neural Inf. Process. System 1601–1608