Chapter two

**Graph mining concepts**

2.0 **What is a Graph?**

A graph provides a convenient way to represent relation between entities and respective data. Each entity is represented by a vertex and their relationship is represented by an edge, where the vertex and/or edges can have arbitrary label [2.1]. A graph ‘G’ composed of two sets: a set of vertices ‘*V*’ and a set of edges ‘ℇ’ can be represented as G = (*V,* ℇ).

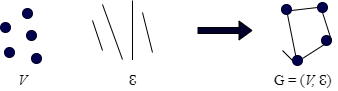


Figure 2.0: Graph

A graph can be undirected or directed.

Undirected graphs are graphs having edges with no direction associated with them. Thus, undirected graphs can be traversed to either direction.

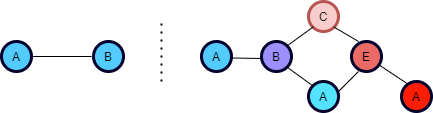


Figure 2.1: Undirected graph example

A directed graph is a graph with set of vertices connected by edges, with each vertex having a direction associated with it.

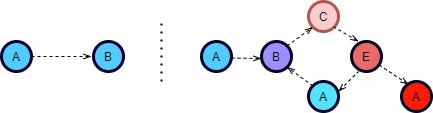


Figure 2.2: Directed graph example

Graph Examples

* Web pages
* Social Networks
* Computer Networks
* Transportation Networks
* Power Networks

2.1 **Graph Mining**

For a graph dataset, in order to extract patterns and gain new knowledge in such a dataset, we need to exploit relational information, this process is called graph mining. New knowledge can be a pattern such as sub graph, an undiscovered relation to other elements, or on a more abstract level expression of trends in data [2.1]. The purpose of graph mining is to discover valuable insights from graph data. For example, in figure 2.3, clustering identifies groups where people behave similarly. Another example is link prediction, which finds pairs of people that are likely to connect with each other.

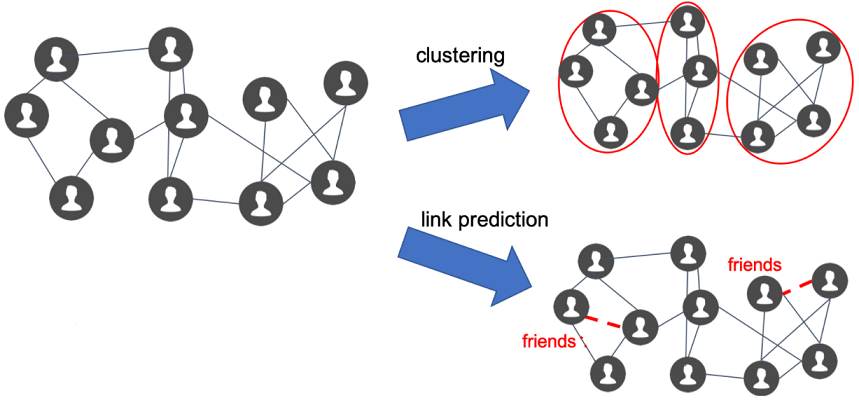


Figure 2.5: Graph mining

**2.2 Applications of graph mining**

**Web graphs**

A web graph is a collection of web pages connected together by hyperlinks. In a web graph, static html pages represent the nodes of the graph while hyperlinks represent the edges [2.18].

Web graphs find application in:

* Providing more accurate search services
* Identifying hubs and authorities
* Computing PageRank of www-pages
* Computing personalized PageRank

**Social Network graphs**

A social network is a network where individuals collaborate. Such a networks are naturally modelled as a graph called social network graph, where individuals represent nodes and the connection between them represents the graph edges [2.19]. Social network graphs can be undirected such as Facebook friends’ graph where individuals connected are both friends to each other or directed, example is the Twitter followers’ graph where person A can follow person B without person B following person A. social network graphs find applications in:

* Identifying the most influential people.
* Recommending friends.
* Companies targeting the right consumers of a product.

**Cybersecurity graphs**

Cybersecurity graphs are graphs where computers represent the graph nodes and message traffic represents the graph edges. Such a graph provides insight of:

* Knowledge of computer viruses’ propagation
* Identifying intruders’ machines
* Predict computers without proper authorization.

**Entertainment graphs**

Entertainment graphs are graphs where actors/movies represent the graph nodes and movies attribute represents the graph edges. Such a graph finds applications in:

* Predicting upcoming movie popularity
* Most likely movies to win award
* Determining most popular movies

2.3 **Graph mining techniques**

**2.3.1 Frequency subgraph mining technique**

Frequency subgraph mining (FSM) is a graph mining technique which discovers subgraphs that occur often in a graph. The discovery of frequent subgraph usually consists of two steps:

* Candidate generation
* Support calculation

The process of discovering the frequent subgraph from the figure 2.4 is as follows: The input to FSM algorithm is a graph dataset G and user defined minimum support(min−sup) and the output is the set of frequent subgraphs S. the support can be calculated my evaluating the number of isomorphs in a given graph [2.2].

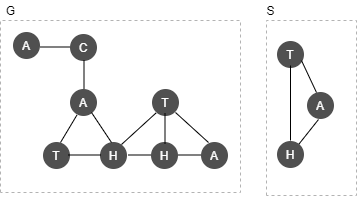


Figure 2.4. The Input Graph G and Subgraphs S of G.

The candidate generation techniques can be classified into two:

* Apriori based approach
* Pattern-growth approach

**Apriori based approach**

The apriori-based approach algorithm works by joining two subgraph which are frequently occurring. The apriori-based approach is similar to frequent item-set mining and it is recursive. The AGM [2.3] algorithm is one of the algorithms for generating candidate graph that increases the substructure size by one vertex at each iteration of the algorithm. Two frequent graphs of size ‘k’ can be merged together only if they have the same (k-1) subgraph. The new candidate subgraph has (k-1) sized component and the additional two vertices [2.1]. A possible problem of this algorithm is the generation of multiple candidates.

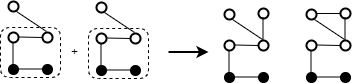
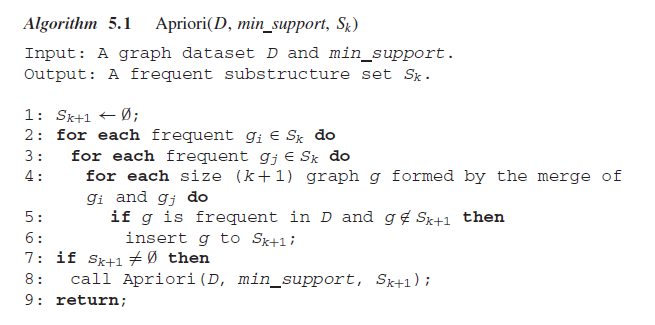


Figure 2.5 AGM



**Pattern based growth approach**

The pattern growth algorithm extends frequent graph by adding new edge, in each possible position. By using this edge-extension method, this method prevents the overhead caused by merging two subgraphs in Apriori approach-based algorithms. However, possible problem of this technique is that same subgraph can be discovered many times [2.1].

**2.3.2 Link Prediction**

The relationship between two individuals in a network could be identified through studying the network and predicting a possible existence of a link between them. The sole aim of link prediction is to identify individuals in the network that will either form a link or not in the future.

The statement of link prediction problem is as follows:

*Given the links in a social network at time t or during a time interval I, we wish to predict the links that will be added to the network during the later time interval from time t’ to a some given future time [2.4].*

Examples of link prediction in a real-world network:

* Friend recommendation in social networks
* Product recommendation in ecommerce
* Interaction prediction in biological networks
* Knowledge graph completion
* Suggesting collaborations between researchers based on co-authorship
* predicting connections between members of terrorist organizations who have not been directly observed to work together.

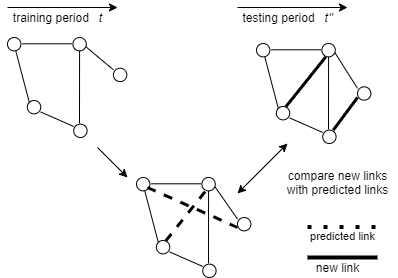


Figure 2.6: Link prediction

Methods of link prediction

* Node neighborhood based
* Ensemble of all path-based

------------

**2.3.3 Classification technique**

Graph classification is a supervised learning problem that focus on calculating certain graph statistics (graph features) that helps in differentiating between graphs of different classes. The statistical features of the graph that the classification is based upon could be the total count of different subgraphs or graphlets [2.5]. For a graph classification task, A set of graphs say {g1,g2,…gn} is given, in which only the labels of a subset of graphs are seen, the goal is to predict the label of unseen graphs. Various methods of graph classification have been developed some of which are:

* Graph classification using neural network
* Kernel graph
* Classification using structural attention

**Kernel Graph**

The kernel method is a machine learning method which measures the similarity of data points which can be used for classification. Different kernel methods exit, the difference between these methods is usually based on how they extract features from graphs. The common kernel methods are: Radom-work graph kernels [2.6], shortest-path graph kernel [2.7], graphlet graph kernels [2.8]. In the random-walk kernel, two graphs ***g1*** and ***g2*** are compared by measuring the random walks of the graphs. Given a graph ***gi*** with vertex *v* ∈ V and edges***lv* ∈ {l1, l2, · · · lk}.** A vector **q(*gi*)** with edges ***li*** and ***lj*** is formed by counting the number of tuples **(*lis , ljd , m*)** in the source vertex ***s*** and destination vertex ***d*** in a walk with m as the length of the walk. In the shortest-path kernel method, the tuple (***lis , ljd , m***) is formed from the shortest path instead of random walk. The graphlet kernels count the number of substructures, graphlets, in the graph.

**Graph classification using neural network**

Two methods usually used for graph classification using neural networks are: Convolutional neural networks (CNNs) and deep learning.

The model developed by Niepert et al [2.9] for CNNs graph classification technique is a case in which a collection of graphs is given, “*the CNNs is required to learn a function that can be used for classification and regression problem of unseen graphs or a large graph is given and the CNNs is required to learn a representation(s) that can be used to infer unseen graph properties such as vertices and edges”.* The CNNs solves this problem by extracting features from local regions (receptive field) of the input done by moving a filter over different regions of the graph.

One of the approaches of graph classification using deep learning developed by Li et al. [2.10] called the DeepGraphs makes use of heat kernels to capture the features from the input graphs needed for the network training. They defined heat kernels over graphs using the eigenvalues and eigenvectors of the normalized graph Laplacian: LN

**2.3.3 Clustering technique**

**https://i11www.iti.kit.edu/\_media/teaching/theses/files/da-huebner-08.pdf**

*Graph clustering (also called graph partitioning) is the process of dividing data in the form of graph. This could be vertex clustering which tends to group nodes of the graph into group of densely related regions based on either edge weights or edge distances or the graph is treated as object to be clustered and cluster these objects on the basis of similarity* [2.11].

Many algorithms used for graph clustering has been developed. These algorithms are broadly divided into global and local techniques based on the input parameter. In global clustering technique, the whole graph is used as input for the clustering process while in local clustering only a certain seed vertex is used. Figure 2.7 shows a grouping of the algorithms into local or global clustering technique [2.12].

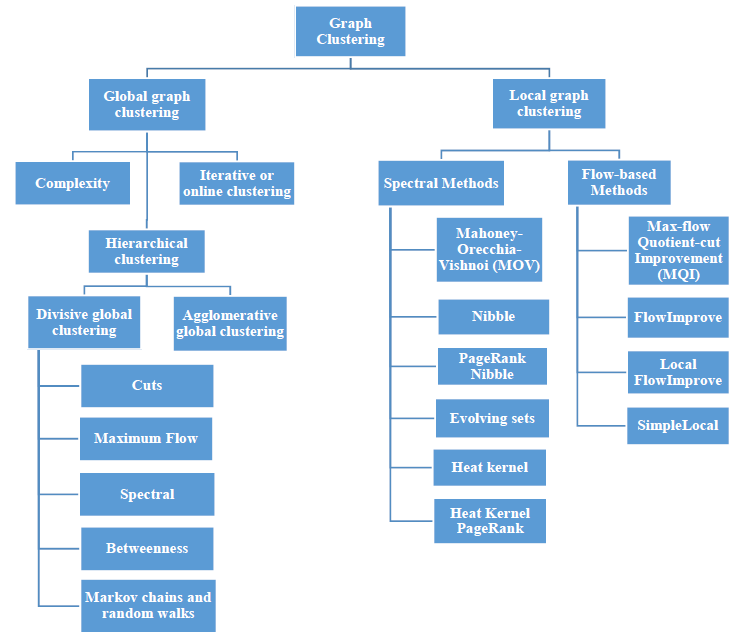


Figure 2.7: Different techniques of Graph Clustering

====

**2.4 Graph mining tools**

There are several tools available for graph mining. In this section, a review of some of the tools currently used for graph mining are discussed.

**2.4.1 Gephi**

Gephi is a modular and extensible open-source Java visualization application built on top of the Netbeans platform. It is suitable for the analysis of all kind of complex networks, and mostly used for social network analysis. Gephi uses OpenGL a cross-language, cross-platform API for 3D rendering of large networks in real-time and to speed up the exploration.

The features of Gephi which makes it a good tool for graph visualization are:

* Real-time visualization: Gephi is powered by OpenGL which makes it a rich tool for real-time visualization of graphs.
* Layout: It has a good layout palette which allows users to change layout settings to increase users experience.
* Metrics: Gephi has a good statistics and metrics framework suitable for social network analysis and scale-free networks.
* Networks over time: In Gephi, users have the ability of manipulating the embedded timeline in which makes them able to visualize how network evolve over time.
* Dynamic filtering: with Gephi interactive user interface, users can filter network in real-time and gives them the ability to select nodes and/or edges based on the network structure.
* Input/Output: Different graph file formats such as CSV and relational databases import are supported by Gephi.
* Extensibility: One of Gephi’s Tools menu items is “Plugins” which makes it possible to import a wide range of community-built plugins to extend its functionalities [2.13].

**2.4.2 NetworkX**

NetworkX is a python language package for exploration and analysis of networks and network algorithms Data structures for representation many types of networks, or graphs. Its flexibility makes it ideal for representing networks large real-world graphs found in many different fields. It good features are:

* Ability to convert graphs to several formats
* Ability to find subgraphs, cliques, k-cores.
* Draw networks in 2D and 3D
* Ability to convert graphs to several formats
* Explore adjacency, degree, diameter, radius, center, betweenness, etc. [2.14]

**2.4.3 Neo4j**

Neo4j is the world’s leading graph database implemented in Java and accessible from software written in other languages using the Cypher query language through a transactional HTTP endpoint, or Bolt protocol. It scales billions of nodes and connections in a system. Its first version was released in 2007 and is available in a GPLv3-licensed open-source “community edition”, with advanced and enterprise versions accessible under AGPLv3.

Neo4j makes use of edges, nodes, or attribute as its data structure. In version 2.0, indexing was added to Cypher with the introduction of schemas [2.15].

**2.4.4 Cytoscape**

Cytoscape is an open-source visualization data mining tool originally developed at the Institute of Systems Biology in Seattle in 2002 for visualizing molecular interaction networks and integrating with gene expression profiles and other state data. The biomedical research community started using this first, and it is useful to understand the gene and protein interaction in biology. Its core features are:

* Support for many standard network and annotation file formats like SIF, GML, XGMML, BioPAX, GraphML etc
* Ability to connect to external public databases and imports network and annotation data.
* Supports for RESTful API for programmatic access
* Support for different image export including PDF, PS, SVG, PNG, JPEG, and BMP files.
* Filter used to select subsets of nodes and/or interaction based on current data [2.16].

**2.4.5 Apache Spark**

Apache Spark is a unified analytics engine first released in 2014 with Apache 2.0 license. It is suitable for large-scale data process, ETL functions, machine learning, as well as cluster computing. The main advantages of Apache Spark are:

* Speed: It has great performance for both streaming and batch data
* Multiple language support (e.g., python, Scala, Java etc)
* Easy to use
* Good cluster management
* Supports for RDDs
* Fault tolerance [2.17]

Reference

[2.1] D. J. Cook and L. B. Holder, Mining Graph Data. Wiley & Sons, 2006

[2.2] Aarzoo Dhiman, S.K. Jain, Optimizing Frequent Subgraph Mining for Single Large Graph, Procedia Computer Science, Volume 89, 2016, Pages 378-385, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2016.06.085.

[2.3] A. Inokuchi, T. Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In Proceedings of 2000 European Symposium Principle of Data Mining and Knowledge Discovery (PKDD’00), pp. 13–23, Lyon, France, Sept. 2000.

[2.4] Al Hasan, Mohammad; Zaki, Mohammed (2011). ["Link Prediction in Social Networks"](http://www.cs.rpi.edu/~zaki/PaperDir/SNDA11.pdf)

[2.5] John Boaz Lee, Ryan Rossi, and Xiangnan Kong. 2018. Graph Classification using Structural Attention. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '18). Association for Computing Machinery, New York, NY, USA, 1666–1674. DOI:https://doi.org/10.1145/3219819.3219980

[2.6] Borgwardt, Karsten M. ; Ong, Cheng S. ; Schönauer, Stefan ; Vishwanathan, SVN ; Smola, Alex J. ; Kriegel, Hans-Peter: Protein function prediction via graph kernels. In: Bioinformatics 21 (2005), No. suppl 1, p. i47–i56

[2.7] Borgwardt, Karsten M.; Kriegel, Hans-Peter: Shortest-path kernels on graphs. In: Data Mining, Fifth IEEE International Conference on IEEE (Veranst.), 2005, p. 8–pp

[2.8] Shervashidze, Nino; Vishwanathan, SVN ; Petri, Tobias ; Mehlhorn, Kurt ; Borgwardt, Karsten M.: Efficient graphlet kernels for large graph comparison. In: AISTATS Bd. 5, 2009, p. 488–495

[2.9] Niepert, Mathias ; Ahmed, Mohamed ; Kutzkov, Konstantin: Learning Convolutional Neural Networks for Graphs. In: arXiv preprint arXiv:1605.05273 (2016)

[2.10] Li, Cheng ; Guo, Xiaoxiao ; Mei, Qiaozhu: DeepGraph: Graph Structure Predicts Network Growth. In: arXiv preprint arXiv:1610.06251 (2016)

[2.11] <https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-30164-8_348>

[2.12] Salem, R.K., Moneim, W.T., & Hassan, M. (2019). GRAPH MINING TECHNIQUES FOR GRAPH CLUSTERING: STARTING POINT.

[2.13] @MISC{Bastian09gephi:an,author={Mathieu Bastian and Sebastien Heymann and Mathieu Jacomy},title={Gephi:An Open Source Software for Exploring and Manipulating Networks},year={2009}}

[2.14] <https://networkx.org/>

[2.15] <https://neo4j.com/product/>

[2.16] <https://cytoscape.org/>

[2.17] <https://spark.apache.org/>

[2.18] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, D. Sivakumar, Andrew Tompkins, and Eli Upfal. 2000. The Web as a graph. In Proceedings of the nineteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. Association for Computing Machinery, New York, NY, USA, 1–10. DOI:https://doi.org/10.1145/335168.335170

[2.19] Jure Leskovec, Anand Rajaraman, & Jeff Ullman. Mining of Massive datasets