Chapter 11
Graph Mining

11.1 Introduction

The contents of the book have focused so far on the mining of data where the underlying structure is characterized by special types of graphs where cycles are not allowed, i.e. acyclic graphs or trees. The focus of this chapter is on the frequent pattern mining problem where the underlying structure of the data can be of general graph type where cycles are allowed. These kinds of representations allow one to model complex aspects of the domain such as chemical compounds, networks, the Web, bioinformatics, etc. Generally speaking, graphs have many undesirable theoretical properties with respect to algorithmic complexity. In the graph mining problem, the common requirement is the systematic enumeration of sub-graphs from a given graph, known as the frequent subgraph mining problem. From the available graph analysis methods, we will narrow our focus to this problem as it is the prerequisite for the detection of interesting associations among graph-structured data objects, and has many important applications. For an extensive overview of graph mining in a general context, including different laws, data generators and algorithms, please refer to (Chakrabati & Faloutsos 2006; Washio & Motoda 2003, Han & Kamber 2006). Due to the existence of cycles in a graph, the frequent subgraph mining problem is much more complex than the frequent subtree mining problem. Even though theoretically it is an NP complete problem, in practice, a number of approaches are very applicable to the analysis of real-world graph data. We will look at a number of different approaches to the frequent subgraph mining problem and a number of approaches for the analysis of graph data in general.

The rest of the chapter is organized as follows. The necessary concepts related to the graph mining problem are discussed in Section 11.2. The graph isomorphism problem is addressed in Section 11.3, which is an important aspect of the frequent subgraph mining process. In Section 11.4, an overview is given of some existing graph mining methods where they have been categorized according to the main underlying approach to the problem. The chapter is concluded in Section 11.5.

F. Hadzic, H. Tan, and T.S. Dillon: Mining of Data with Complex Structures, SCI 333, pp. 287–300.
springerlink.com

11.2 General Graph Concepts and Definitions

A graph is a set of nodes where each node can be connected to another node including itself. There are many types of graphs, such as directed/undirected, weighted, finite/infinite, regular and complete graphs. Often, these types exist for specialized applications where the specific relationships or constraint hold, or are to be enforced to hold. However, most semi-structured documents where the underlying structure is a graph, can be modeled as a general graph with undirected and unlabeled edges. In that sense, a graph can be denoted as $G = (V,L,E)$, where (1) V is the set of vertices or nodes; (2) L is a labelling function that assigns a label $L(v)$ to every vertex $v \in V$; and (3) $E = \{(v_1,v_2) | v_1,v_2 \in V\}$ is the set of edges in G. The number of nodes in G (i.e. $|V|$) is called the *order* of G while the number of edges ($|E|$) is referred to as the *size* of G. Each edge usually has two nodes associated with it, but it is also possible that only a single node is associated with it in the case when there is an edge from one node to itself (i.e. a cycle). If two vertices v_1 and v_2 are connected by an edge, then they are said to be *adjacent* to one another, and *nonadjacent* or *independent*, otherwise. Two edges that meet at a vertex are said to be *adjacent*, and *nonadjacent* otherwise. In other words, if two edges (v_{a1}, v_{a2}) and (v_{b1}, v_{b2}) are adjacent and ($v_{a1} \neq v_{a2}$) and ($v_{b1} \neq v_{b2}$), then one of the following holds: ($v_{a1} = v_{b1}$) or ($v_{a1} = v_{b2}$) or ($v_{a2} = v_{b1}$) or ($v_{a2} = v_{b2}$). The set of all vertices adjacent to a vertex v corresponds to the neighbourhood of v. A *path* is defined as a finite sequence of edges between any two nodes and, as opposed to a tree where there is a single unique path between any two nodes, in a graph there could be multiple paths. The *length of a path p* is the number of edges in p. The fan-out/degree of a node is the number of edges emanating from that node.

A graph G' is a subgraph of another graph G if there exists a subgraph isomorphism between G' and G. Isomorphism is a one-to-one and onto mapping from the node set of one graph to the node set of another where the adjacency and non-adjacency is preserved.

As is the case with trees, there are a number of different types of subgraphs and some of the most common ones in the context of frequent subgraph mining are discussed next together with a formulization of the subgraph isomorphism problem.

11.3 Graph Isomorphism Problem

Two graphs $G_1(V_1,L_1,E_1)$ and $G_2(V_2,L_2,E_2)$ are said to be isomorphic to one another if there exists a structure preserving vertex bijection $f: V_1 \rightarrow V_2$ such that $(v_1, v_2) \in E_1$ iff $(f(v_1), f(v_2)) \in E_2$. Hence, two labeled graphs $G_1(V_1,L_1,E_1)$ and $G_2(V_2,L_2,E_2)$ are isomorphic to each other if there is one-to-one mapping from V_1 to V_2 that preserves vertices, labels of vertices and adjacency and non-adjacency of the vertices.

As mentioned earlier, a graph is a subset of another graph if there exists a subgraph isomorphism between them. More formally, this can be stated as follows:

A graph G_S (V_S,L_S,E_S) is a *subgraph* of graph G (V,L,E) iff $V_S \subseteq V$ and $E_S \subseteq E$.

The frequent subgraph mining problem can be generally stated as: Given a database of graphs G_{DB} and a minimum support threshold (σ), extract all subgraphs that occur at least σ times in G_{DB}.

In addition to the general subgraph definition provided above, other common subgraph types are spanning, connected and induced subgraphs.

A graph G_S (V_S,L_S,E_S) is a *spanning* subgraph of graph G (V,L,E) iff $V_S = V$ and $E_S \subseteq E$.

A graph G_S (V_S,L_S,E_S) is a *connected* subgraph of graph G (V,L,E) iff $V_S \subseteq V$ and $E_S \subseteq E$, and all vertices in V_S are mutually reachable through some edges in E_S.

A graph G_S (V_S,L_S,E_S) is an *induced* subgraph of graph G (V,L,E) iff $V_S \subseteq V$, $E_S \subseteq E$, and there exists a mapping $f{:}V_S \rightarrow V$ such that for any pair of vertices v_x and $v_y \in V_S$ if there is an edge $(f(v_x), f(v_y)) \in E \Leftrightarrow (v_x, v_y) \in E_S$.

To illustrate these aspects, please consider Fig. 11.1 where we show an example graph database G_{DB} consisting of three transactions. The different types of subgraphs of transaction 1 are shown in Fig. 11.2, and some frequent (general) subgraphs for support 2 and 3 are shown in Figure. 11.3.

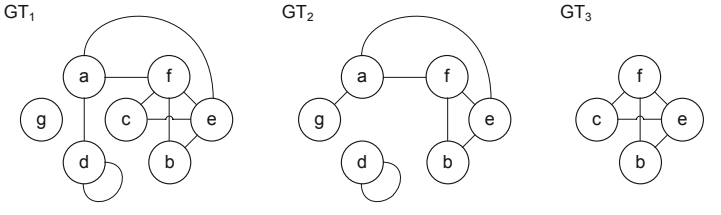

Fig. 11.1 Example graph database G_{DB} consisting of 3 transactions GT_1, GT_2 and GT_3

In this section, we narrowed our focus to the most commonly used subgraph definitions within the frequent pattern mining problem from graph-structured data in the data mining field. There are quite a few different variations of graphs considered in the large research problem of analysis of graph-structured data, as well as different types of measures and constraints that can be imposed on the analysis task. For details about these aspects and other related issues, we refer the interested reader to (Chakrabati & Faloutsos 2006, Washio & Motoda 2003). In the next section, we consider a variety of methods developed for the frequent graph mining problem. In some of these methods, constraints have been imposed to either orient the analysis better toward a specific application aim, or to decrease the number of patterns enumerated to alleviate the complexity problem introduced by mining graph-structured data.

11.4 Existing Graph Mining Methods

A number of graph-based data mining methods have been developed. In this section, we will focus mainly on the methods developed for solving the frequent subgraph

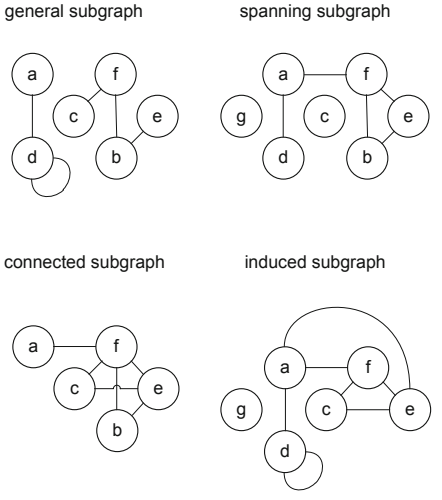

Fig. 11.2 Example subgraphs of graph representing transaction 1 (GT_1) from graph database G_{DB} of Fig. 11.1

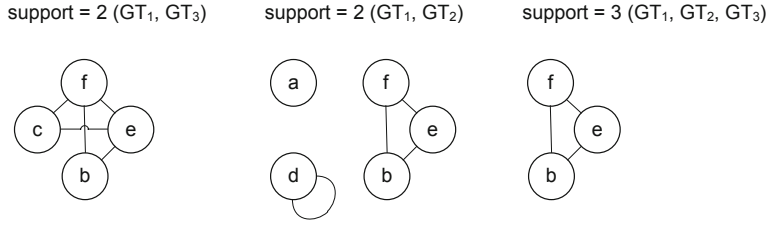

Fig. 11.3 Some frequent general subgraphs from graph database G_{DB} of Fig. 11.1

mining task explained earlier in Section 11.3. At the end of the section we overview a number of methods developed for solving related problems to frequent subgraph mining and the analysis of graph data in general.

11.4.1 Apriori-Like Methods

This section focuses on the graph mining methods which were developed using the same principle of the Apriori-based frequent itemset/sequence/subtree mining. The candidate enumeration process is performed in a bottom-up manner, starting with subgraphs consisting of one node (i.e. 1-subgraphs). At each iteration, the candidate k-subgraphs are checked for frequency and only the frequent patterns are used for generating (k+1)-subgraphs. There are a number of variations among the Apriori-based algorithms in regards to the way the candidates are generated. As was discussed earlier in the book, the join approach that works so well for the frequent itemset mining may not be so suitable for application to cases where the structural

properties of data patterns need to be taken into account. Many candidates that are invalid are unnecessarily generated and pruned. In frequent subgraph mining, this is even more so the case, due to the many ways in which the join operation on two substructures can be performed.

The AGM algorithm (Inokuchi, Washio & Motoda, 2000) adopts the level-wise Apriori approach where graphs are represented using an adjacency matrix. Starting from the smallest subgraphs, the candidates are enumerated by performing the join operation on the adjacency matrices representing candidate subgraphs. A canonical form for induced subgraphs is defined according to which each adjacency matrix needs to be sorted so that the frequency can be correctly determined. The FSG algorithm (Kuramochi & Karypis 2001), was developed to extract frequent connected undirected subgraphs. To minimize storage and computational processing, FSG uses a sparse graph representation to effectively store input transactions, candidate and frequent subgraphs. They perform canonical form labelling of adjacency matrix and then convert it to adjacency-list representation. To generate candidate (k+1)-subgraphs from the frequent k-subgraphs, the join operation is performed on the frequent k-subgraphs that contain the same (k-1)-subgraph. The frequency of the newly generated (k+1)-subgraph is determined as the size of the intersection of the transactional identifier lists (TID lists) of the joined k-subgraphs. The use of the TID lists simplifies the candidate enumeration and counting process in FSG, but the storage requirements of all the TID lists for large graph data may cause memory problems. The same authors (Kuramochi & Karypis 2002), have proposed the gFSG algorithm which is an extension to the problem of finding frequently occurring geometric patterns in geometric graphs. These geometric graphs are graphs whose vertices have two or three dimensional coordinates associated with them. The gFSG also uses a level-wise approach extending frequent subgraphs by one edge at a time, and a number of algorithms are integrated for computing the isomorphism between geometric subgraphs, that are rotation, scaling and translation invariant.

Another Apriori-based algorithm has been proposed in (Vanetik, Gudes & Shimony, 2002). It uses canonical representations of paths and path sequences, and a lexicographical ordering over path pairs is defined based on node labels and degrees of nodes within paths. A graph is represented as a union of *edge-disjoint paths*, where two paths are edge-disjoint if they do not share any common edge. A bottom-up approach is used where firstly, all frequent subgraphs consisting of single path are found and these are combined, where possible, to construct subgraphs consisting of two paths. The algorithm progressively enumerates subgraphs consisting of k paths, by joining frequent subgraphs with k-1 paths.

11.4.2 Pattern-Growth Methods

The commonality among the algorithms that adopt an Apriori-like approach is that subgraphs are systematically enumerated in a bottom-up manner where problems can occur when the data is too large or the structural relationships are fairly complex. This is especially the case when the candidates are generated using the join approach. There are many possibilities where two subgraphs can be joined, and the

structural variations of a candidate subgraph may not always exist in the database. The subgraph isomorphism is an expensive test, and hence, generating candidates which then need to be pruned is wasteful. These problems have motivated the development of a number of algorithms which adopt more of a graph-structure-guided approach to minimize the unnecessary candidate subgraphs generated.

The gSpan algorithm (Yan & Han, 2002) was the first approach that adopts a depth-first based search for frequent subgraphs. The search is supported by a novel canonical labelling system. Each graph is mapped to a sequential code and all codes are sorted according to the ascending lexicographical order. A depth-first search is then applied to the trees matching the first nodes of the codes and progressively adding more frequent descendants. This subgraph increase stops when the support of a graph becomes less than the minimum support or when it has already been discovered earlier in the method. The gSpan algorithm is very efficient in terms of the time taken and the memory required.

The algorithm presented in (Borgelt & Berthold 2002), focuses on the discovery of frequent substructures in molecular compounds and also uses a depth-first search strategy to find frequent subgraphs. The occurrence of a fragment within all molecules is kept in parallel during the pattern (fragment) growth process. The local order of the atoms and bonds is used to prune the unnecessary fragment extensions which speeds up the search and avoids the generation of redundant patterns. The FFSM algorithm (Huan, Wang & Prins 2003) uses an algebraic graph framework for an unambiguous enumeration of frequent subgraphs. Each graph is represented using an adjacency matrix and the subgraph isomorphism is avoided by exploiting the embeddings of each frequent subgraph. The GASTON (GrAph/Sequence/Tree extractiON) algorithm (Nijssen & Kok 2004) has been developed with the underlying idea of a quick start for search of frequent graph structures by integrating the search for frequent paths, trees and graphs into one approach. A level-wise approach is used that first enumerates simple paths, followed by tree structures and the most complex graph structures at the end. This kind of approach is motivated by the observation that in practice, most graphs are not actually so complex and the number of cycles is not too large. Hence, in the enumeration phase, sequences/paths are enumerated first and then tree structures are enumerated by repeatedly adding one node and an incident edge to the nodes in the path. Graph structures are then enumerated by adding edges in between the nodes of the tree structure.

11.4.3 Inductive Logic Programming (ILP) Methods

The approaches that fall within this category are characterized by the use of the ILP to represent graph data using horn clauses. They come from the multi relational data mining field, which is essentially the mining of data that is organized in multiple interlinked tables of a relational database.

The WARMR algorithm (Dehaspe & Toivonen 1999) has been developed for frequent query extraction from a DATALOG database. Compared with standard frequent pattern mining methods, an extra parameter (*key*) is used, which is essentially an attribute that must be contained in all of the patterns extracted. It allows the user

to restrict the search space to only those patterns that contain the item of interest. The algorithm is flexible in the types of patterns that can be extracted. A declarative language bias (WRMODE) is formulized that restricts the search space to only the admissible and potentially interesting queries. The main processes of the algorithm are candidate generation and candidate evaluation. The candidate evaluation step is used to determine the frequency of the generated candidates. In the candidate generation step, the patterns that contain the specified key are found and extended incrementally, i.e. starting from the smallest patterns. At each step, the frequent and infrequent patterns are determined and the frequent patterns are extended by adding one node at a time. The patterns are pruned if they are either already contained in the frequent pattern set, or if they are a specialization of a pattern contained in the infrequent pattern set. The method was applied to the analysis of telecommunication alarm and chemical toxicology. The major shortcoming of the algorithm is its efficiency since the equivalence test between first-order clauses is very complex. This has motivated Nijssen & Kok (2001) to propose an alternative solution for this expensive step. Their algorithm FARMER still uses the first order logic notation and is by and large very similar to WARMR, with a substantial time performance gain. The main difference is that instead of the expensive equivalence tests, the FARMER utilizes a tree data structure to both count and generate candidate queries in an efficient manner. De Raedt & Kramer (2001) developed a method to find frequent molecular fragments from a molecular compound database. A molecular fragment is defined as a sequence of linearly connected atoms, and the database contains information of the elements of the atoms of a molecule and bond orders. The approach allows the specification of a generality constraint so that all the found molecular fragments satisfy a conjunction of primitive constraints. In addition to the traditionally used minimum frequency parameter, their approach allows the use of the maximum frequency constraint on patterns. These constraints allow more flexibility in the types of queries that can be answered through the patterns, and in the molecular fragment finding domain, more interesting patterns could be found. Another ILP-based approach has been presented in (Lisi & Malerba 2004) and a hybrid language is proposed to retain information regarding the relational as well as structural properties in the multi-level association rules mined from multiple relations. Query subsumption relationships are utilized to define a generality order and a downward refinement operator.

11.4.4 Greedy Search Methods

The characteristics of the greedy search-based methods are that they avoid excessive computation required for the search of all frequent subgraphs, and hence use a greedy approach to reduce the number of subgraphs considered. The high complexity of the graph isomorphism problem is avoided at the cost of missing some frequent subgraphs.

One of the first graph mining approaches is known as the SUBDUE system (Cook & Holder 1993) and many refinements and optimizations of this system were made (Cook & Holder 2000; Cook et al. 2001; Jonyer, Holder & Cook 2002; Noble

& Cook 2003; Holder et al. 2003; Ketkar, Holder & Cook 2005). The SUBDUE system is based on the minimum description length (MDL) principle, measured as the number of bits necessary to completely describe the graph. Instances of discovered subgraphs are replaced by concept definitions. This compresses the original dataset and provides a basis for discovering hierarchically-defined structures. The motivation of this kind of approach is to identify conceptually interesting substructures that enhance the interpretation of the data. Besides using the MDL principle, the SUBDUE system allows for the incorporation of other background knowledge to focus the search on more suitable subgraphs. A greedy beam search approach is used to discover candidate subgraphs from the data. It starts from single nodes and iteratively expands the instances of substructures with one neighboring edge at a time to cover all possible expansions. Each newly generated candidate subgraph is then expanded and the algorithm considers all possible substructures in the search for the best substructures according to the minimum description length. The algorithm stops when either all possible substructures have been considered or the computational limit has been reached. An optional pruning technique is used to avoid expansions of those substructures whose description length would increase. In its search for matching substructures, SUBDUE utilizes an inexact graph match algorithm to allow for slight variations since interesting substructures may often show up in a slightly different form throughout the data.

Similarly, the Graph Based Induction (GBI) method (Yoshida, Motoda & Indurkhya 1994) compresses the graph data in order to arrive at interesting subgraphs with minimal size. It was initially developed to find interesting concepts from frequent patterns found in the inference trace. The compression technique used is the so called pair-wise chunking, where two nodes are paired (chunked) into one node. The links between paired nodes are removed and if necessary the links to other nodes in the graph are 'remembered' so that at any time during the search, the original graph can be reconstructed. The pair-wise chunking can be nested and the graph can be repeatedly compressed. A minimal size is chosen to limit the amount of compression which reflects the sizes of extracted patterns and the compressed graph. The search for local subgraph candidates is performed using an opportunistic beam search.

11.4.5 Other Methods

The methods discussed in the previous sections all belong to the family of frequent substructure (subgraph) mining methods. The amount of research that has gone into automatic analysis of graph data in general is enormous and for a broad overview of the different graph mining laws, constraints, specialized applications and algorithms please refer to (Chakrabarti & Faloutsos, 2006; Washio & Motoda, 2003, Han & Kamber 2006). This section will mention some alternative approaches to graph data mining, which are usually motivated by the different data analysis aims or specific application needs or constraints.

Clustering of graph-structured data is generally important in many applications, as discovered clusters often provided the basis for analysis of similarities and differences, and can be used to classify graph-structured data objects based on the characteristics of the formed clusters. For example, the graph clustering technique based on the network flow theory has been applied to the tissue segmentation of magnetic resonance images of the human brain in (Wu & Leahy 1993). The data is represented as an undirected adjacency edge where each edge is assigned a flow capacity indicating the similarity of the nodes connected by the edge. Clusters are formed by the continuous removal of edges from the graph until mutually exclusive subgraphs are formed. The edges are removed based on their flow capacity with the aim of minimizing the largest maximum flow (similarity) among the formed subgraphs (clusters). Mancoridis et al. (1998) have developed a number of clustering techniques for the automatic recovery of the modular structure of a software system from its source code. The source code is represented as a module dependency graph and a combination of clustering, hill-climbing and genetic algorithms are used to discover the high level structure of the systems organization.

A number of graph-clustering algorithms are focused on optimizing specific clustering criteria that occur in the context of graph clustering, and often borrow methods from the more general graph optimization problems. For example, a graph theory-based cluster analysis approach, has been presented in (Hartuv & Shamir 2000). A similarity graph is defined and clusters are defined as subgraphs with connectivity that is above half the number of vertices, and the algorithm has low polynomial complexity. The graph-clustering algorithm presented in (Flake, Tarjan and Tsioutsiouliklis 2004) is based on the general idea of maximum flow techniques to maximize intra-cluster similarity and minimize inter-cluster similarity. An artificial sink is placed in the graph that is connected to all the nodes, and the maximum flows are calculated between all the nodes and the sink. A parameter is chosen to limit the number of edges used in connecting the artificial sink to the other nodes of the graph. The clustering is based on minimum cut trees. A minimal cut tree is a subgraph of the original graph where the path between two given nodes in the minimal cut tree, is guaranteed to be the smallest path between those two nodes in the original graph. Hence, by selecting minimum cut trees from the expanded graphs, the algorithm discovers quality clusters and heavily connected components. A kernel function between two graphs has been proposed in (Kashima, Tsuda & Inokuchi 2003). The graphs are represented as a feature vector by counting the *label paths* that appear in the graph. The label paths are produced by random walk functions on the graph, and the kernel is defined as the inner product of the feature vectors averaged over all possible label paths. The computation of the kernel becomes the problem of finding the stationary state of a discrete-time linear system, and the solution becomes that of solving simultaneous linear equations with a sparse coefficient matrix.

Most of the clustering approaches just described will not completely solve the frequent subgraph mining problem as defined in Section 11.3, as it is not guaranteed that the discovered subgraphs using clustering approaches will contain all the frequent subgraphs for a given support. Hence, they provide an approximate solution, and can often overcome some of the complexity limitations that occur when

all frequent subgraphs need to be completely enumerated. This approximation can be useful in certain applications, as often, discovering all the frequent subgraphs for a given support will result in too many patterns that are not of interest for the application at hand,

Another approximate frequent subgraph mining method based on spanning trees has been proposed in (Zhang, Yang & Cheedella 2007). As mentioned in Section 11.3, a spanning subgraph of a graph must contain all its vertices. A spanning tree is a spanning subgraph that is a tree (i.e. no cycles). In other words, a spanning tree from an undirected, connected graph G, is a selection of edges from G that span every vertex from G, with no cycles. An example spanning tree is shown in Fig. 11.4. The Monkey algorithm (Zhang, Yang & Cheedella 2007), is motivated by the observation that in cases of edge distortion, one needs to allow for some flexibility in the graph isomorphism checking in order to detect these frequent patterns in spite of the variation on edges. This flexibility in the isomorphism checking is introduced by searching for frequent spanning trees since, in a spanning subtree, all the vertices must be present while some of the edges can be absent as long as the subgraph is still a tree. Monkey is based on depth-first search and two constraints are integrated to allow one to control the unrelated edges in affecting the frequency of corresponding patterns.

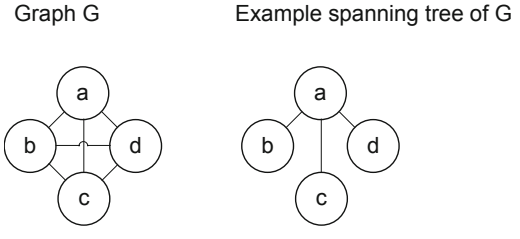

Fig. 11.4 An example spanning tree of a graph G

The problem of mining large disk-based graph databases has been addressed in (Wang et al. 2004). The main contribution is the *ADI* (adjacency index) structure to support major tasks in graph mining when the graph databases cannot be held in main memory. The previously discussed gSpan algorithm (Yan & Han, 2002) was adapted to use the *ADI* structure, and the resulting algorithm *ADI-Mine* is shown to outperform gSpan when large disk-based graph databases are in question. The ADI-Mine algorithm and a number of constraint-based mining techniques have been integrated together to form the GraphMiner system (Wang et al. 2005). The system provides a graphical user interface so that a number of constraints can be selected, such as: pattern size constraints, the edges/vertices that must appear or not appear in the patterns, graphs of which the patterns must be super- or sub-patterns and aggregate constraints. Furthermore, the system allows for easy browsing and analysis of patterns and querying capabilities to focus on the application-specific or interesting patterns.

Saigo & Tsuda (2008) have proposed an iterative subgraph mining method for principal component analysis, where salient patterns are progressively collected by separate graph mining calls. At each graph mining call, real-value weights are assigned to graph transactions, and patterns satisfying the weighted support threshold are enumerated. The search strategy is based on a branch-and-bound algorithm that uses the depth-first search code tree as the canonical search space. The patterns are organized as trees, such that a child node has a supergraph of the pattern in its parent node. Patterns are enumerated by systematically generating patterns from the root to the leaves in a recursive manner using right-most-node expansion.

11.4.6 Mining Closed/Maximal Subgraph Patterns

Similar to other frequent pattern mining problems, the mining of frequent closed and maximal subgraphs is an important area of research as it is one of the ways to reduce the complexity problem introduced by enumerating all of the frequent subgraphs.

The CloseGraph algorithm (Yan & Han 2003), mines closed frequent subgraphs and its overall framework is similar to the gSpan algorithm (Yan & Han 2002) with the addition of suitable graph pruning techniques to effectively reduce the search space and enumerate only closed subgraphs. The algorithm starts by pruning all infrequent nodes and edges, and then generates all frequent graphs consisting of a single node. It then extends the set of frequent graphs recursively using the depth-first search and right-most extension. The depth-first search is based on the DFS lexicographical order, i.e the novel canonical labelling system first introduced in their gSpan algorithm. The authors define an early termination condition based upon the equivalence of occurrence of frequent subgraphs in the original graph, so that not all the graphs need to be extended and checked for satisfying the closed subgraph property. This early termination condition is not valid for all subgraphs and another condition is imposed to detect such cases in order to avoid loss of information regarding potentially frequent close subgraphs. These conditions allow the CloseGraph algorithm to detect conditions in which all of descendant subgraphs of a frequent graph cannot be closed and need not be checked.

The problem of mining frequent closed subgraphs with connectivity constraints in relational graphs, has been addressed in (Yan, Zhou & Han 2005), and two algorithms, CloseCut and Splat, have been proposed. The CloseCut algorithm is a pattern-growth approach, i.e. small frequent subgraphs are first enumerated and extended by adding new edges. The candidate subgraphs generated by this process must satisfy the defined connectivity constraints and the minimum frequency threshold. The candidate graphs are extended by adding new edges until the resulting graph after edge addition is no longer frequent. The Splat algorithm, on the other hand, is a pattern-reduction based approach, whereby the relational graphs are intersected and decomposed in order to obtain highly connected graphs. At each step, the algorithm checks whether a newly generated graph exists in the results set, in which case it needs to no longer be processed, since no closed highly connected graphs can be enumerated from that candidate graph.

The MARGIN algorithm (Thomas, Valluri & Karlapalem 2006) mines frequent maximal subgraphs in a top-down manner. For each graph record of a database of graphs, the algorithm repetitively drops one edge at a time, without generating disconnected graphs, until a frequent representative of that graph record is found. This frequent representative is likely to be a frequent maximal pattern once a number of infrequent edges and nodes have been removed. Hence, a recursive procedure is called on the generated frequent connected subgraphs, which progressively cuts infrequent edges and nodes from the given subgraph until it satisfies the properties of a frequent maximal subgraph.

11.5 Conclusion

This chapter has given a brief introduction to the problem of graph mining, which is not restricted to acyclic graphs which is the main focus of this book. Some formal definitions were given with an illustrative example to allow the reader to gain some understanding of the complexity of the problem and some of the undesirable properties causing this. This complexity also explains the reason behind so many different approaches to solving the graph mining problem as was discussed in Section 11.4. We can see that the novelty often comes when the problem is alleviated to some extent such as is the case in GASTON algorithm (Nijssen & Kok 2004) which gets a quick start by first discovering simpler patterns and increasing the complexity by moving from sequence, trees and finally to cyclic graphs. However, it is worth mentioning that even though the graph mining problem appears theoretically infeasible, in practical situations the existing algorithms can often complete the task within a reasonable amount of time, on real-world graph-structured data. Applications of graph mining are manifold and in general they are useful for analysis of data in any domain, given that the data objects are organized in a graph structure. When the algorithms were discussed previously, we mentioned a few application areas, and we can generally state that the applications are somewhat similar to the many applications of tree mining discussed in Chapter 9, with the main difference being that in graph mining the application data can contain cycles among the data objects. Some other example applications are chemical compound analysis, social network analysis, web structure mining, ontology mining, and in general graph mining techniques are useful for many web and social-media applications.

References

1. Bern, M., Eppstein, D.: Approximation Algorithms For Geometric Problems. In: Hochbaum, D.S. (ed.) Approximation Algorithms for NP-Hard Problems, pp. 296–345. PWS Publishing Company (1996)
2. Borgelt, C., Berthold, M.R.: Mining Molecular Fragments: Finding Relevant Substructures of Molecules. Paper presented at the Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM), Maebashi City, Japan, December 9-12 (2002)
3. Chakrabarti, D., Faloutsos, C.: Graph mining: Laws, Generators and Algorithms. ACM Computing Surveys 38(1), 2-es (2006)

4. Cook, D.J., Holder, L.B.: Substructure Discovery Using Minimum Description Length and Background Knowledge. Journal of Artificial Intelligence Research 1(1), 231–255 (1993)
5. Cook, D.J., Holder, L.B.: Graph-Based Data Mining. IEEE Transactions on Intelligent Systems 15(2), 32–41 (2000)
6. Cook, D.J., Holder, L.B., Galal, G., Maglothin, R.: Approaches to Parallel Graph-Based Knowledge Discovery. Journal of Parallel and Distributed Computing 61(3), 427–446 (2001)
7. De Raedt, L., Kramer, S.: The levelwise version space algorithm and its application to molecular fragment finding. Paper presented at the Proceedings of the 17th International Joint Conference on Artificial intelligence, Seattle, WA, USA, August 4-10 (2001)
8. Dehaspe, L., Toivonen, H.: Discovery of frequent DATALOG patterns. Data Mining and Knowledge Discovery 3(1), 7–36 (1999)
9. Flake, G.W., Tarjan, R.E., Tsioutsiouliklis, K.: Graph Clustering and Minimum Cut Trees. Internet Mathematics 1(4), 385–408 (2004)
10. Han, J., Kamber, M.: Data Mining: Concepts and Techniques, 2nd edn. Elsevier, Morgan Kaufmann Publishers, San Francisco, CA, USA (2006)
11. Hartuv, E., Shamir, R.: A Clustering Algorithm Based on Graph Connectivity. Information Processing Letters 76(4-6), 175–181 (2000)
12. Holder, L.B., Cook, D.J., Djoko, S.: Substructure Discovery in the SUBDUE System. Paper presented at the Proceedings of the AAAI Workshop on Knowledge Discovery in Databases, Seattle, Washington, USA, July 31- August 4 (1994)
13. Holder, L., Cook, D., Gonzalez, J., Jonyer, I.: Structural Pattern Recognition in Graphs. In: Chen, D., Chen, X. (eds.) Pattern Recognition and String Matching, pp. 255–279. Kluwer Academic Publishers, Dordrecht (2003)
14. Huan, J., Wang, W., Prins, J.: Efficient mining of frequent subgraph in the presence of isomorphism. Paper presented at the Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003), Melbourne, Florida, USA, December 19-22 (2003)
15. Inokuchi, A., Washio, T., Motoda, H.: An apriori-based algorithm for mining frequent substructures from graph data. Paper presented at the Proceedings of the 4th European Conference on Principles and Practice of Knowledge Discovery in Databases, Lyon, France, September 13-16 (2000)
16. Jonyer, I., Holder, L.B., Cook, D.J.: Graph-based hierarchical conceptual clustering. Journal of Machine Learning Research 2, 19–43 (2002)
17. Kashima, H., Tsuda, K., Inokuchi, A.: Marginalized kernels between labeled graphs. Paper presented at the Proceedings of the 20th International Conference on Machine Learning (ICML 2003), Washington, DC, USA, August 21-24 (2003)
18. Ketkar, N.S., Holder, L.B., Cook, D.J.: Subdue: compression-based frequent pattern discovery in graph data. Paper presented at the Proceedings of the ACM SIGKDD 1st International Workshop on Open source Data Mining, Chicago, Illinois, USA, August 21-24 (2005)
19. Kuramochi, M., Karypic, G.: Frequent Subgraph Discovery. Paper presented at the Proceedings of the IEEE International Conference on Data Mining (ICDM 2001), San Jose, California, USA, November 29 - December 2 (2001)
20. Kuramochi, M., Karypis, G.: Discovering Frequent Geometric Subgraphs. Paper presented at the Proceedings of the 2nd IEEE International Conference on Data Mining (ICDM 2002), Maebashi City, Japan, December 9-12 (2002)
21. Lisi, F.A., Malerba, D.: Inducing Multi-Level Association Rules from Multiple Relations. Machine Learning 55(2), 175–210 (2004)

22. Mancoridis, S., Mitchell, B., Rorres, C., Chen, Y., Gansner, E.: Using Automatic Clustering to Produce High-Level System Organizations of Source Code. Paper presented at the Proceedings of the 6th International Workshop on Program Comprehension (IWPC 1998), Los Alamitos, CA, USA, June 26 (1998)
23. Nijssen, S., Kok, J.N.: A Quickstart in Frequent Structure Mining Can Make a Difference. Paper presented at the Proceedings of the, International Conference on Knowledge Discovery and Data Mining (KDD 2004), Seattle, WA, USA, August 22-25 (2004)
24. Noble, C.C., Cook, D.J.: Graph-based anomaly detection. Paper presented at the Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24-27 (2003)
25. Saigo, H., Tsuda, K.: Iterative Subgraph Mining for Principal Component Analysis. Paper presented at the Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), Pisa, Italy, December 15-19 (2008)
26. Thomas, S., Sarawagi, S.: Mining Generalized Association Rules and Sequential Patterns using SQL Queries. In: Proc. 4th Intl. Conf. on Knowledge Discovery and Data Mining (KDD 1998), pp. 344–348 (1998)
27. Vanetik, N., Gudes, E., Shimony, S.E.: Computing Frequent Graph Patterns from Semistructured Data. Paper presented at the Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), Maebashi City, Japan, December 9-12 (2002)
28. Wang, C.W., Pei, J., Zhu, Y., Shi, B.: Scalable Mining of Large Disk-Based Graph Databases. Paper presented at the Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, WA, USA, August 22-25 (2004)
29. Wang, W., Wang, C., Zhu, Y., Shi, B., Pei, J., Yan, X., Han, J.: GraphMiner: a structural pattern-mining system for large disk-based graph databases and its applications. Paper presented at the Proceedings of the, ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, USA, June 14-16 (2005)
30. Washio, T., Motoda, H.: State of the art of graph-based data mining. ACM SIGKDD Explorations Newsletter 5(1), 59–68 (2003)
31. Wilson, R., Hancock, E., Luo, B.: Pattern vectors from algebraic graph theory. IEEE Transactions on Pattern Analysis and Machine Intelligence 27(7), 1112–1124 (2005)
32. Yan, X., Han, J.: gSpan: Graph-based substructure pattern mining. Paper presented at the Proceedings of the, IEEE International Conference on Data Mining (ICDM), Maebashi City, Japan, December 9-12 (2002)
33. Yan, X., Han, J.: CloseGraph: mining closed frequent graph patterns. In: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24-27, pp. 286-295 (2003)
34. Yan, X., Zhou, X.J., Han, J.: Mining Closed Relational Graphs with Connectivity Constraints. Paper presented at the Proceedings of the 11th ACM SIGKDD International Cofnerence on Knowledge Discovery and Data Mining (KDD 2005), Chicago, Illinois, USA, August 21-24 (2005)
35. Yoshida, K., Motoda, H., Indurkhya, N.: Graph-based induction as a unified learning framework. Journal of Applied Intelligence 4(3), 297–316 (1994)
36. Zhang, S., Yang, J., Cheedella, V.: Monkey: Approximate Graph Mining Based on Spanning Trees. Paper presented at the Proceedings of the IEEE 23rd International Conference on Data Engineering (ICDE 2007), Istanbul, Turkey, April 15-20 (2007)