REVIEW OF LITERATURE

2. REVIEW OF LITERATURE

Frequent patterns are itemsets, subsequences, or substructures that appear in a data set with frequency more than a user-specified threshold. A set of items that appear frequently together in a transaction data set, is a frequent itemset. A subsequence, if it occurs frequently in a shopping history database, is a (frequent) sequential pattern. A substructure can refer to different structural forms, such as subgraphs, subtrees, or sublattices, which may be combined with itemsets or subsequences. If a substructure occurs frequently in a graph database, it is called a (frequent) structural pattern.

Frequent pattern mining has been a focused theme in data mining research for over a decade. Abundant literature has been dedicated to this research and tremendous progress has been made, ranging from efficient and scalable algorithms for frequent itemset mining in transaction databases to numerous research frontiers, such as sequential pattern mining (Liu *et al.*, 2002), structured pattern mining (Asai *et al.*, 2002; Zaki, 2002), correlation mining (Brin *et al.*, 2007), associative classification (Borgelt, 2003), and frequent pattern-based clustering (Inokuchi *et al.*, 1999), as well as their broad applications. In this chapter, a brief overview of the current status of frequent pattern mining is provided both in relation to transactional database and web log data.

2.1. BASIC FREQUENT PATTERN MINING ALGORITHMS

The most frequently used basic frequent itemset mining methodologies are Apriori, FP-growth and Eclat. Since there are usually a large number of distinct single items in a typical transaction database, and their combinations may form a very huge number of itemsets, it is challenging to develop scalable methods for mining frequent itemsets in a large transaction database. Agrawal and Srikant (1994) observed an interesting downward closure property, called Apriori, among frequent k-itemsets: A k-itemset is frequent only if all of its sub-itemsets are frequent. This implies that frequent itemsets can be mined by

first scanning the database to find the frequent 1-itemsets, then using the frequent 1-itemsets to generate candidate frequent 2-itemsets, and check against the database to obtain the frequent 2-itemsets. This process iterates until no more frequent k-itemsets can be generated for some k. This is the essence of the Apriori algorithm and its alternative (Mannila *et al.* 1994).

From the point of introduction, there have been extensive studies on the improvements or extensions of Apriori, e.g., hashing technique (Park *et al.*, 1995), partitioning technique (Savasere *et al.*, 1995), sampling approach (Toivonen, 1996), dynamic itemset counting (Brin *et al.*, 1997), incremental mining (Cheung *et al.*, 1996), parallel and distributed mining (Agrawal and Shafer, 1996; Zaki *et al.*, 1997), and integrating mining with relational database systems (Sarawagi *et al.*, 1998). Geerts *et al.* (2001) derived a tight upper bound of the number of candidate patterns that can be generated in the level-wise mining approach. This result is effective at reducing the number of database scans.

In many cases, the Apriori algorithm significantly reduces the size of candidate sets using the Apriori principle. However, it can suffer from two-nontrivial costs: (1) generating a huge number of candidate sets, and (2) repeatedly scanning the database and checking the candidates by pattern matching. Han *et al.* (2000) devised an FP-growth method that mines the complete set of frequent itemsets without candidate generation.

FP-growth works in a divide-and-conquer way. The first scan of the database derives a list of frequent items in which items are ordered by frequency descending order. According to the frequency-descending list, the database is compressed into a frequent-pattern tree, or FP-tree, which retains the itemset association information. The FP-tree is mined by starting from each frequent length-1 pattern (as an initial suffix pattern), constructing its conditional pattern base (a "subdatabase", which consists of the set of prefix paths in the FP-tree co-occurring with the suffix pattern), then constructing its

conditional FP-tree, and performing mining recursively on such a tree. The pattern growth is achieved by the concatenation of the suffix pattern with the frequent patterns generated from a conditional FP-tree.

The FP-growth algorithm transforms the problem of finding long frequent patterns to searching for shorter ones recursively and then concatenating the suffix. It uses the least frequent items as a suffix, offering good selectivity. Performance studies demonstrate that the method substantially reduces search time.

There are many alternatives and extensions to the FP-growth approach, including depth-first generation of frequent itemsets by Agarwal *et al.* (2001); H-Mine, by Pei *et al.* (2001) which explores a hyper-structure mining of frequent patterns; building alternative trees; exploring top-down and bottom-up traversal of such trees in pattern-growth mining by Liu *et al.* (2002; 2003); and an array-based implementation of prefix-tree-structure for efficient pattern growth mining by Grahne and Zhu (2003).

Both the Apriori and FP-growth methods mine frequent patterns from a set of transactions in horizontal data format (i.e., {TID: itemset}), where TID is a transaction-id and itemset is the set of items bought in transaction TID. Alternatively, mining can also be performed with data presented in vertical data format (i.e., {item:TID_set}).

Zaki (2000) proposed EquivalenceCLAssTransformation (Eclat) algorithm by exploring the vertical data format. The first scan of the database builds the TID_set of each single item. Starting with a single item (k = 1), the frequent (k+1)-itemsets grown from a previous k-itemset can be generated according to the Apriori property, with a depth-first computation order similar to FP-growth (Han *et al.* 2000). The computation is done by intersection of the TID_sets of the frequent k-itemsets to compute the TID_sets of the corresponding (k+1)-itemsets. This process repeats, until no frequent itemsets or no candidate itemsets can be found.

Besides taking advantage of the Apriori property in the generation of candidate (k + 1)-itemset from frequent k-itemsets, another merit of this method is that there is no need to scan the database to find the support of (k+1)-itemsets (for k $\geq$1). This is because the TID_set of each k-itemset carries the complete information required for counting such support.

Another related work which mines the frequent itemsets with the vertical data format is (Holsheimer *et al.* 1995). This work demonstrated that, though impressive results have been achieved for some data mining problems using highly specialized and clever data structures, one could also explore the potential of solving data mining problems using the general purpose database management systems (DBMS).

Apart from these three algorithms, several other algorithms are also used for frequent pattern mining. They are summarized in Table 2.1.

TABLE 2.1

TRADITIONAL FREQUENT PATTERN MINING ALGORITHMS

Algorithms	Year	Description
candidate-generation-and-test approach algorithms		
Apriori[5]	1994	The most basic algorithm in candidate-generation-and-test approach
Partition[56]	1995	Apriori variant which reduces the number of TDB scan
SEAR[39]	1995	Apriori variant which reduce the size of memory usage by storing itemsets to prefix-tree instead of hash-tree
DIC[52]	1997	Apriori variant which reduce the number of TDB scan
DHP[44]	1998	Apriori variant which reduce the number of candidate itemsets
pattern-growth approach algorithms		
FP-growth[21]	2000	Extracting itemsets with original data structure, called FP-tree
H-Mine[33]	2001	Efficient algorithm for sparse dataset by using original dataset, called H-struct
PD[45]	2001	Extracting itemset with removing infrequent itemsets from original dataset
FP-growth*[18]	2003	Improved FP-growth for sparse dataset

2.2. APPLICATIONS OF FREQUENT PATTERN MINING

Frequent patterns, reflecting strong associations among multiple items or objects, capture the underlying semantics in data. They were successfully applied to inter-disciplinary domains beyond data mining. Among several applications of frequent pattern mining, four important ones, namely, indexing and similarity search, multimedia data mining, mining data streams, web mining are discussed in this section.

2.2.1. Indexing and Similarity Search

Complex objects such as transaction sequence, event logs, proteins and images are widely used in many fields. Efficient search of these objects becomes a critical problem for many applications. Due to the large volume of data, it is inefficient to perform a sequential scan on the whole database and examine objects one by one. High performance indexing mechanisms thus are in heavy demand in filtering objects that obviously violate the query requirement. Yan *et al.* (2004) proposes a discriminative frequent pattern-based approach to index structures and graphs. In order to find those queries which only have infrequent patterns this method replaced the uniform support constraint with a size-increasing support function, which has very low support for small patterns but high support for large patterns. This concept behind this model can also be applied to indexing sequences, trees, and other complicated structures as well. SeqIndex is one example using frequent pattern-based approach to index sequences. Taking frequent patterns as features, new strategies to perform structural similarity search were developed such as Grafil (Yan *et al.* 2005) and PIS (Yan *et al.* 2006).

2.2.2. Spatiotemporal database and Multimedia data mining

A spatial database stores a large amount of space-related data, such as maps, preprocessed remote sensing or medical imaging data, and VLSI chip layout data. A spatiotemporal database stores time-related spatial data, such as

weather dynamics, moving objects, or regional developments. Spatial data mining refers to the extraction of knowledge, spatial relationships, or other interesting patterns from spatial data. Similarly, spatiotemporal data mining is to find spatiotemporal knowledge and patterns. Due to the complexity of spatiotemporal data objects and their relationships as well as their associated high computational cost, it is costly to mine spatiotemporal frequent patterns in spatiotemporal data.

One important methodology that may substantially reduce the computational cost is progressive refinement (Koperski and Han, 1995), which performs rough computation at a coarse resolution and refines the results only for those promising candidates at finer resolutions. They proposed a methodology for mining spatial association rules (or frequent patterns). At the coarse level of resolution, a rough spatial approximation such as minimum bounding rectangles can be used to estimate the frequent pattern candidates. Then, only those frequent pairs will need to be re-examined at finer levels of resolution using more refined but expensive spatial computation.

Similar methodology has been used in mining co-location patterns, by Xiong *et al.* (2004) and Zhang *et al.* (2004), where further optimization is performed by considering the spatial co-location property, i.e., the spatially closely located objects usually have more interesting and closer relationships than the objects located far apart. Such optimization ideas can be extended to mining spatiotemporal sequential patterns as well, as shown in Cao *et al.* (2005).

Moreover, Li *et al.* (2006) show that even for mining outliers in massive moving object data sets, one can find movement fragment patterns by spatial overlay, and such movement fragments can be taken as motifs for further identification of outliers by motif-based classification.

A multimedia database system stores and manages a large collection of multimedia data, such as audio, video, image, graphics, speech, text, document,

and hypertext data. Multimedia data mining is finding patterns and knowledge from multimedia data. Frequent pattern analysis in multimedia data plays a similar important role in multimedia data mining. To mine frequent patterns in multimedia data, each image object can be treated as a transaction and frequently occurring patterns among different images can be discovered. An image may contain multiple objects, each with many features such as color, shape, texture, keyword, and spatial location, so there could be many possible associations. Moreover, since a picture containing multiple recurrent objects is an important feature in image analysis, recurrence of the same object should be considered as important in frequent pattern analysis. Furthermore, spatial relationships among different objects in an image are also considered crucial in image analysis. Thus all these factors will be considered in multimedia frequent pattern mining. Zaïane *et al.* (2000) takes those factors into consideration and developed a progressive refinement algorithm for mining multimedia associations.

2.2.3. Mining data streams

Tremendous and potentially infinite volumes of data streams are often generated by real-time surveillance systems, communication networks, Internet traffic, online transactions in the financial market or retail industry, electric power grids, industry production processes, scientific and engineering experiments, remote sensors, and other dynamic environments. Unlike traditional data sets, stream data flow in and out of a computer system continuously and with varying update rates. It may be impossible to store an entire data stream or to scan through it multiple times due to its tremendous volume. To discover knowledge or patterns from data streams, it is necessary to develop single-scan and on-line mining methods.

For mining frequent items and itemsets on stream data, Manku and Motwani proposed sticky sampling and lossy counting algorithms for approximate frequency counts over data streams. Karp *et al.* (2003) proposed a

counting algorithm for finding frequent elements in data streams. Chang and Lee (2003) proposed an algorithm for finding recent frequent itemsets adaptively over an online data stream by decaying the effect of old transactions.

Yu *et al.* (2004) proposed an FDPM algorithm for mining frequent itemsets over data streams with a false-negative oriented approach. It is argued that, compared with the false-positive mining approach (e.g., lossy counting), the false-negative approach can effectively mine frequent itemsets with a bound of memory consumption, while in the false-positive approach, the number of false-positive frequent itemsets could increase exponentially, which makes the mining intractable. Chi *et al.* (2004) proposed an algorithm for mining closed frequent itemsets over a sliding window. A synopsis data structure is designed to monitor the transactions in the sliding window and a compact data structure, a closed enumeration tree is used to maintain a dynamically selected set of itemsets.

Metwally *et al.* (2005) proposed a method for computing frequent and top-k elements in data streams by carefully using buffer space. Jin and Agarwal (2005) proposed a one pass algorithm for frequent itemset mining which has deterministic bound on the accuracy and does not require any out-of-core summary structure. Lin *et al.* (2005) proposed an algorithm for mining frequent itemsets from data streams based on a time-sensitive sliding window.

2.2.4. Web mining applications

Web mining is the application of data mining techniques to discover patterns and knowledge from the Web (Kosala and Blockeel 2000; Srivastava *et al.* 2000). There are three different types of web mining: web content mining, web structure mining, and web usage mining. Web content mining is a knowledge discovery task of finding information within web pages, while web structure mining aims to discover knowledge hidden in the structures linking web pages. Web usage mining is focused on the analysis of users' activities

when they browse and navigate through the Web. Classical examples of web usage mining include, but not limited to, user grouping (users that often visit the same set of pages), page association (pages that are visited together), and sequential clickthrough analysis (the same browse and navigation orders that are followed by many users). Association rules discovered for pages that are often visited together can reveal user groups (Eirinaki and Vazirgiannis 2003) and cluster web pages. Web access patterns via association rule mining in web logs were proposed by (Chen *et al.* 1996; Pei *et al.* 2000; Srivastava *et al.* 2000; Punin and Krishnamoorthy 2001). Sequential pattern mining in web logs could find browse and navigation orders (i.e., pages that are accessed immediately after another), which might be used to refine cache design and web site design. More complicated patterns such as frequent tree-like traversal patterns were examined by (Chen *et al.* 1996; Nanopoulos and Manolopoulos 2001).

2.3. WEB USAGE MINING AND FREQUENT PATTERN MINING

In Web usage mining, association rules are used in order to discover the pages which are visited together even if they are not directly connected, which can reveal associations between group of users with specific interest. This information can be used for restructuring Web sites by adding links between those pages which are visited together. Association rules in Web logs are discovered by Chen *et al.* (1996), Punin *et al.* (2001), Batista *et al.* (2002), Zaiane *et al.* (1998), Shen *et al.* (2000).

2.3.1. Web Log Mining and Apriori Algorithm

Association rule generation can be used to relate pages that are most often referenced together in a single server session. In the context of Web Usage Mining, association rules refer to sets of pages that are accessed together with a support value exceeding some specified threshold. These pages may not be directly connected to one another via hyperlinks. For example, association rule discovery using the Apriori algorithm (or one of its variants) may reveal a

correlation between users who visited a page containing electronic products to those who access a page about sporting equipment. Aside from being applicable for business and marketing applications, the presence or absence of such rules can help Web designers to restructure their Web site. The association rules may also serve as a heuristic for prefetching documents in order to reduce user-perceived latency when loading a page from a remote site.

There are many other examples where association rules have been used, for example users' visits of WWW pages, in which the structure and its content can be optimized. Baowen (2001) have used re-ranking method and generalized Association Rules to extract access patterns of the Web sites pattern usage.

Mannila *et al.*, (1994) use page accesses from a Web server log as events for discovering frequent episodes. The major data mining technique used in their research work was association rules.

Chen *et al.*, (1996) introduce the concept of using the maximal forward references in order to break down user sessions into transactions for the mining of traversal patterns.

Batista and Silva, (2001) perform mining process for online newspaper Web access logs by using Apriori algorithm. Apriori was the first scalable algorithm designed for association-rule mining algorithm. Apriori is an improvement over the AIS and SETM algorithms.

The Apriori algorithm searches for large itemsets during its initial database pass and uses its result as the seed for discovering other large datasets during subsequent passes. Rules having a support level above the minimum are called large or frequent itemsets and those below are called small itemsets (Chen *et al.*, 1996).

Zorrilla *et al.* (2005) used association knowledge to discover knowledge from web logs and recommended their system for online applications such as

web recommendation and personalization. Their experiments showed that the rules generated are comparable in quality.

Yu *et al.* (2001) proposed a novel incremental mining algorithm using FP-growth algorithm for identifying frequent patterns of web usage mining. Their results were comparable with the existing standards and had the potential to a web prototype of Web Log Analyzer in web usage mining.

Wang *et al.* (2004) proposed a method that can discover users' frequent access patterns underlying users' browsing Web behaviors using association rules. They proposed a technique which used a revised algorithm (FAP-Mining) based on the FP-tree algorithm to mine frequent access patterns. The algorithm is accurate and scalable for mining frequent access patterns with different lengths.

2.3.2. Web Log Mining and FP-Growth Algorithm

Another approach to associative web log mining is the use of FP-growth algorithm. According to Sun and Zhang (2004) mining frequent patterns from Web logs is an important data mining task and candidate-generation-and-test and FP-growth are two representative frequent pattern mining approaches. They conducted extensive experiments on real world Web log data to analyze the characteristics of Web logs and the behaviour of these two approaches on Web logs. To improve the performance of FP-Growth algorithm on mining Web logs, they proposed a new algorithm, namely, Combined Frequent Pattern Mining (CFPM) to cater for Web log data specifically.

Dong *et al.* (2005) used a combination of neural networks and FP-Growth algorithm to discover frequent patterns from web log files. In the same period, Li *et al*, (2005) used FP-growth algorithm to discover the path traversal patterns over web log click sequences.

A detailed review of frequent pattern discovery in web log data is discussed by Iváncsy and Vajk (2006). Dong *et al.* (2007) presented a novel

incremental mining algorithm of frequent patterns for web usage mining which used FP-Growth algorithm during frequent pattern mining.

Romero *et al.* (2009) applied FP-growth algorithm for personalizing hyperlinks in Web-based adaptive educational systems. Similarly, Sun and Zhao (2009) designed and implemented an E-learning model based on web usage mining Techniques built on FP-Growth. Recently, Kumar and Rukmani (2010) discovered web usage patterns using Apriori and FP-Growth algorithms.

2.4. OTHER FREQUENT MINING TECHNIQUES

Apart from market basket database and web log files, frequent pattern mining techniques have been used in many other situations. These techniques are summarized in this section.

2.4.1. Mining Multilevel, Multidimensional and Quantitative Association Rules

Since data items and transactions are (conceptually) organized in multilevel and/or multidimensional space, it is natural to extend mining frequent itemsets and their corresponding association rules to multi-level and multidimensional space. Multilevel association rules involve concepts at different levels of abstraction, whereas multidimensional association rules involve more than one dimension or predicate.

In many applications, it is difficult to find strong associations among data items at low or primitive levels of abstraction due to the sparsity of data at those levels. On the other hand, strong associations discovered at high levels of abstraction may represent commonsense knowledge. Therefore, multilevel association rules provide sufficient flexibility for mining and traversal at multiple levels of abstraction. Multilevel association rules can be mined efficiently using concept hierarchies under a support-confidence framework. For example, if the min_sup threshold is uniform across multi-levels, one can first mine higher-level frequent itemsets and then mine only those itemsets

whose corresponding high-level itemsets are frequent (Han and Fu 1995). Moreover, redundant rules can be filtered out if the lower-level rules can essentially be derived based on higher-level rules and the corresponding item distributions (Srikant and Agrawal 1995). Efficient mining can also be derived if min_sup varies at different levels (Han and Kamber 2006). Such methodology can be extended to mining multidimensional association rules when the data or transactions are located in multidimensional space, such as in a relational database or data warehouse (Kamber *et al.* 1997).

All the above mentioned techniques work on discrete items, such as item name, product category, and location. For finding frequent patterns and associations for numerical attributes, such as salary, age, and scores, quantitative association rules can be mined, with a few alternative methods. In this method, the data was mined in binned intervals and then clustering mined quantitative association rules for concise representation as in the ARCS system, by Lent *et al.* (1997); based on x-monotone and rectilinear regions by Fukuda *et al.* (1996) and Yoda *et al.* (1997), or using distance-based (clustering) method over interval data, by Miller and Yang (1997).

Mining quantitative association rules based on a statistical theory to present only those that deviate substantially from normal data was studied by Aumann and Lindell (1999). Zhang *et al.* (2004) considered mining statistical quantitative rules. Statistical quantitative rules are quantitative rules in which the right hand side of a rule can be any statistic that is computed for the segment satisfying the left hand side of the rule.

2.4.2. Mining Closed and Maximal Frequent Itemsets

A major challenge in mining frequent patterns from a large data set is the fact that such mining often generates a huge number of patterns satisfying the min_sup threshold, especially when min_sup is set low. This is because if a pattern is frequent, each of its sub-patterns is frequent as well. A large pattern will contain an exponential number of smaller, frequent sub-patterns. To

overcome this problem, closed frequent pattern mining and maximal frequent pattern mining were proposed.

A pattern α is a closed frequent pattern in a data set D if α is frequent in D and there exists no proper super-pattern β such that β has the same support as α in D. A pattern α is a maximal frequent pattern (or max-pattern) in set D if α is frequent, and there exists no super-pattern β such that $\alpha \subset \beta$ and β is frequent in D. For the same min_sup threshold, the set of closed frequent patterns contains the complete information regarding to its corresponding frequent patterns; whereas the set of max-patterns, though more compact, usually does not contain the complete support information regarding to its corresponding frequent patterns.

The mining of frequent closed itemsets was proposed by Pasquier *et al.* (1999), where an Apriori-based algorithm called A-Close for such mining was presented. Other closed pattern mining algorithms include CLOSET (Pei *et al.* 2000), CHARM (Zaki and Hsiao 2002), CLOSET+ (Wang *et al.* 2003), FPClose (Grahne and Zhu 2003) and AFOPT (Liu *et al.* 2003). The main challenge in closed (maximal) frequent pattern mining is to check whether a pattern is closed (maximal).

There are two strategies to approach this issue: (1) to keep track of the TID list of a pattern and index the pattern by hashing its TID values. This method is used by CHARM which maintains a compact TID list called a diffset; and (2) to maintain the discovered patterns in a pattern-tree similar to FP-tree. This method is exploited by CLOSET+, AFOPT and FPClose. A Frequent Itemset Mining Implementation (FIMI) workshop dedicated to the implementation methods of frequent itemset mining was reported by Goethals and Zaki (2003). Mining closed itemsets provides an interesting and important alternative to mining frequent itemsets since it inherits the same analytical power but generates a much smaller set of results. Better scalability and interpretability is achieved with closed itemset mining.

Mining max-patterns was first studied by Bayardo (1998), where MaxMiner an Apriori-based, level-wise, breadth-first search method was proposed to find max-itemset by performing superset frequency pruning and subset infrequency pruning for search space reduction. Another efficient method MAFIA, proposed by Burdick *et al.* (2001), uses vertical bitmaps to compress the transaction id list, thus improving the counting efficiency. Yang (2004) provided theoretical analysis of the (worst-case) complexity of mining max-patterns. The complexity of enumerating maximal itemsets is shown to be NP-hard. Ramesh *et al.* (2003) characterized the length distribution of frequent and maximal frequent itemset collections. The conditions are also characterized under which one can embed such distributions in a database.

2.4.3. Mining High-Dimensional Datasets

The growth of bioinformatics has resulted in datasets with new characteristics. Micro array and mass spectrometry technologies, which are used for measuring gene expression level and cancer research respectively, typically generate only tens or hundreds of very high-dimensional data (e.g., in 10,000 – 100,000 columns). If each sample is taken as a row (or TID) and each gene as a column (or item), the table becomes extremely wide in comparison with a typical business transaction table. Such datasets pose a great challenge for existing (closed) frequent itemset mining algorithms, since they have an exponential number of combinations of items with respect to the row length.

Pan *et al.* (2003) proposed CARPENTER, a method for finding closed patterns in high-dimensional biological datasets, which integrates the advantages of vertical data formats and pattern growth methods. By converting data into vertical data format {item: TID_set}, the TID_set can be viewed as rowset and the FP-tree so constructed can be viewed as a row enumeration tree. CARPENTER conducts a depth-first traversal of the row enumeration tree, and checks each rowset corresponding to the node visited to see whether it is frequent and closed.

Pan *et al.* (2004) proposed COBBLER, to find frequent closed itemset by integrating row enumeration with column enumeration. Its efficiency has been demonstrated in experiments on a data set with high dimension and a relatively large number of rows.

Liu *et al.* (2006) proposed TD-Close to find the complete set of frequent closed patterns in high dimensional data. It exploits a new search strategy, top-down mining, by starting from the maximal row set, integrated with a novel row enumeration tree, which makes full use of the pruning power of the min_sup threshold to cut down the search space. Furthermore, an effective closeness-checking method is also developed that avoids scanning the dataset multiple times.

2.4.4. Mining Sequential Patterns

A sequence database consists of ordered elements or events, recorded with or without a concrete notion of time. There are many applications involving sequence data, such as customer shopping sequences, Web clickstreams, and biological sequences. Sequential pattern mining, the mining of frequently occurring ordered events or subsequences as patterns has become an important problem in data mining.

Generalized Sequential Patterns (GSP), a representative Apriori-based sequential pattern mining algorithm, proposed by Srikant and Agrawal (1996), uses the downward-closure property of sequential patterns and adopts a multiple pass, candidate generate-and-test approach. GSP also generalized their earlier notion in Agrawal and Srikant (1995) to include time constraints, a sliding time window, and user-defined taxonomies. Zaki (2001) developed a vertical format-based sequential pattern mining method called SPADE, which is an extension of vertical format-based frequent itemset mining methods, like Eclat and CHARM(Zaki 1998; Zaki and Hsiao 2002). In vertical data format, the database becomes a set of tuples of the form itemset (sequence_ID, event_ID). The set of ID pairs for a given itemset forms the ID_list of the

itemset. To discover the length-k sequence, SPADE joins the ID_lists of any two of its length-(k − 1) subsequences. The length of the resulting ID_list is equal to the support of the length-k sequence. The procedure stops when no frequent sequences can be found or no sequences can be formed by such joins. The use of vertical data format reduces scans of the sequence database. The ID_lists carry the information necessary to compute the support of candidates. However, the basic search methodology of SPADE and GSP is breadth-first search and Apriori pruning. Both algorithms have to generate large sets of candidates in order to grow longer sequences.

PrefixSpan, a pattern-growth approach to sequential pattern mining, was developed by Pei *et al.* (2001). PrefixSpan works in a divide-and-conquer way. The first scan of the database derives the set of length-1 sequential patterns. Each sequential pattern is treated as a prefix and the complete set of sequential patterns can be partitioned into different subsets according to different prefixes. To mine the subsets of sequential patterns, corresponding projected databases are constructed and mined recursively.

A performance comparison of GSP, SPADE, and PrefixSpan shows that PrefixSpan has the best overall performance (Pei *et al.* 2004). SPADE, although weaker than PrefixSpan in most cases, outperforms GSP. The comparison also found that when there is a large number of frequent subsequences, all three algorithms run slowly. The problem can be partially solved by closed sequential pattern mining, where closed subsequences are those sequential patterns containing no supersequence with the same support. The CloSpan algorithm for mining closed sequential patterns was proposed by Yan *et al.* (2003). The method is based on a property of sequence databases, called equivalence of projected databases, stated as follows: Two projected sequence databases, $S|\alpha = S|\beta$, $\alpha \_ \beta$, are equivalent if and only if the total number of items in $S|\alpha$ is equal to the total number of items in $S|\beta$, where $S|\alpha$ is the projected database with respect to the prefix α. Based on this property, CloSpan can prune the non-closed sequences from further consideration during

the mining process. A later algorithm called BIDE, a bidirectional search for mining frequent closed sequences was developed by Wang and Han (2004), which can further optimize this process by projecting sequence datasets in two directions.

The studies of sequential pattern mining have been extended in several different ways. Mannila *et al.* (1997) consider frequent episodes in sequences, where episodes are essentially acyclic graphs of events whose edges specify the temporal before-and-after relationship but without timing-interval restrictions. Sequence pattern mining for plan failures was proposed in Zaki *et al.* (1998). Garofalakis *et al.* (1999) proposed the use of regular expressions as a flexible constraint specification tool that enables user-controlled focus to be incorporated into the sequential pattern mining process. The embedding of multidimensional, multilevel information into a transformed sequence database for sequential pattern mining was proposed by Pinto *et al.* (2001). Pei *et al.* (2002) studied issues regarding constraint-based sequential pattern mining. CLUSEQ is a sequence clustering algorithm, developed by Yang and Wang (2003). An incremental sequential pattern mining algorithm, IncSpan, was proposed by Cheng *et al.* (2004). SeqIndex, efficient sequence indexing by frequent and discriminative analysis of sequential patterns, was studied by Cheng *et al.* (2005).

A method for parallel mining of closed sequential patterns was proposed by Cong *et al.* (2005). A method, MSPX, for mining maximal sequential patterns by using multiple samples, was proposed by Luo and Chung (2005). Data mining for periodicity analysis has been an interesting theme in data mining. Özden *et al.* (1998) studied methods for mining periodic or cyclic association rules. Lu *et al.* (1998) proposed inter transaction association rules, which are implication rules whose two sides are totally ordered episodes with timing-interval restrictions (on the events in the episodes and on the two sides).

Bettini *et al.* (1998) consider a generalization of inter transaction association rules. The notion of mining partial periodicity was first proposed by Han, Dong, and Yin, together with a max-subpattern hit set method (Han *et al.* 1999). Ma and Hellerstein (2001) proposed a method for mining partially periodic event patterns with unknown periods. Yang *et al.* (2003) studied mining asynchronous periodic patterns in time-series data. Mining partial order from unordered 0-1 data was studied by Gionis *et al.* (2003) and Ukkonen *et al.* (2005). Pei *et al.* (2005) proposed an algorithm for mining frequent closed partial orders from string sequences.

2.4.5. Mining Structural Patterns: Graphs, Trees and Lattices

Many scientific and commercial applications need patterns that are more complicated than frequent itemsets and sequential patterns. Such sophisticated patterns go beyond sets and sequences, toward trees, lattices, and graphs. As a general data structure, graphs have become increasingly important in modeling sophisticated structures and their interactions, with broad applications including chemical informatics, bioinformatics, computer vision, video indexing, text retrieval, and Web analysis.

Among the various kinds of graph patterns, frequent substructures are the very basic patterns that can be discovered in a collection of graphs. Recent studies have developed several frequent substructure mining methods. Washio and Motoda (2003) conducted a survey on graph-based data mining. Holder *et al.* (1994) proposed SUBDUE to do approximate substructure pattern discovery based on minimum description length and background knowledge. Dehaspe *et al.* (1998) applied inductive logic programming to predict chemical carcinogenicity by mining frequent substructures. Besides these studies, there are two basic approaches to the frequent substructure mining problem: an Apriori-based approach and a pattern-growth approach.

Apriori-based approach

Apriori-based frequent substructure mining algorithms share similar characteristics with Apriori-based frequent itemset mining algorithms. The search for frequent graphs starts with graphs of small "size", and proceeds in a bottom-up manner. At each iteration, the size of newly discovered frequent substructures is increased by one. These new substructures are first generated by joining two similar but slightly different frequent subgraphs that were discovered already.

The frequency of the newly formed graphs is then checked. Typical Apriori-based frequent substructure mining algorithms include AGM by Inokuchi *et al.* (1998), FSG by Kuramochi and Karypis (2001), and an edge disjoint path-join algorithm by Vanetik *et al.* (2002). The AGM algorithm uses a vertex-based candidate generation method that increases the substructure size by one vertex at each iteration. Two size-k frequent graphs are joined only when the two graphs have the same size-(k−1) subgraph. Here, graph size means the number of vertices in a graph. The newly formed candidate includes the common size-(k − 1) subgraph and the additional two vertices from the two size-k patterns. Because it is undetermined that whether there is an edge connecting the additional two vertices, AGM actually can form two candidates.

The FSG algorithm adopts an edge-based candidate generation strategy that increases the substructure size by one edge in each iteration. Two size-k patterns are merged if and only if they share the same subgraph having k-1 edges. In the edge-disjoint path method, graphs are classified by the number of disjoint paths they have, and two paths are edge-disjoint if they do not share any common edge. A substructure pattern with k+1 disjoint paths is generated by joining substructures with k disjoint paths.

The Apriori-based algorithms mentioned above have considerable overhead when two size-k frequent substructures are joined to generate size-(k+1) graph candidates. In order to avoid such overhead, non-Apriori-based

algorithms have been developed, most of which adopt the pattern-growth methodology, as discussed below.

Pattern-growth approach

Pattern-growth-based graph pattern mining algorithms include gSpan by Yan and Han (2002), MoFa by Borgelt and Berthold (2002), FFSM by Huan *et al.* (2003), SPIN by Huan *et al.* (2004), and Gaston by Nijssen and Kok (2004). These algorithms are inspired by PrefixSpan (Pei *et al.* 2001), TreeMinerV (Zaki 2002), and FREQT (Asai *et al.* 2002) atmining sequences and trees, respectively.

The pattern-growth mining algorithm extends a frequent graph by adding a new edge, in every possible position. A potential problem with the edge extension is that the same graph can be discovered many times. The gSpan algorithm solves this problem by introducing a right-most extension technique, where the only extensions take place on the right-most path. A right-most path is the straight path from the starting vertex v0 to the last vertex vn, according to a depth-first search on the graph.

Besides the frequent substructure mining algorithms, constraint-based substructure mining algorithms have also been proposed. Mining closed graph patterns was studied by Yan and Han (2003) with the proposal of the algorithm, CloseGraph, as an extension of gSpan and CloSpan (Yan *et al.* 2003).

Mining coherent subgraphs was studied by Huan *et al.* (2004). For mining relational graphs, Yan *et al.* (2005) proposed two algorithms, CloseCut and Splat, to discover exact dense frequent substructures in a set of relational graphs. For large-scale graph database mining, a disk-based frequent graph mining method was proposed by Wang *et al.* (2004). Jin *et al.* (2005) proposed an algorithm, TSMiner, for mining frequent large-scale structures (defined as topological structures) from graph datasets. Techniques are developed for pushing constraints and specifying approximate matches. Kuramochi and

Karypis (2004) proposed an algorithm, GREW, for finding patterns corresponding to connected subgraphs that have a large number of vertex-disjoint embeddings from a large graph. Ting and Bailey (2006) proposed an algorithm for mining the minimal contrast subgraph which is able to capture the structural differences between any two collections of graphs.

2.5. MINING COMPRESSED FREQUENT PATTERNS

To reduce the huge set of frequent patterns generated in data mining while maintain the high quality of patterns, recent studies have been focusing on mining a compressed or approximate set of frequent patterns. In general, pattern compression can be divided into two categories: lossless compression and lossy compression, in terms of the information that the result set contains, compared with the whole set of frequent patterns.

Mining closed patterns, is a lossless compression of frequent patterns. Mining all non-derivable frequent sets proposed by Calders and Goethals (2002) belongs to this category as well since the set of result patterns and their support information generated from these methods can be used to derive the whole set of frequent patterns. A depth-first algorithm, based on Eclat, was proposed by Caldersand Goethals (2005) for mining the non-derivable itemsets.

Liu *et al.* (2006) proposed to use a positive border with frequent generators to form a lossless representation. Lossy compression is adopted in most other compressed patterns, such as maximal patterns by Bayardo (1998), top-k most frequent closed patterns by Wang *et al.* (2005), condensed pattern bases by Pei *et al.* (2002), k-summarized patterns or pattern profiles by Afrati *et al.* (2004) and Yan *et al.* (2005), and clustering-based compression by Xinetal. (2005).

Forming top-k most frequent closed patterns, a TFP algorithm (Wang *et al.* 2005) is proposed to discover top-k closed frequent patterns of length not less than min_l. TFP gradually raises the support threshold during the mining

and prunes the FP-tree both during and after the tree construction phase. Due to the uneven frequency distribution among itemsets, the top-k most frequent patterns usually do not represent the most representative k patterns. Another branch of the compression work takes a "summarization" approach where the aim is to derive k representatives which cover the whole set of (closed) frequent itemsets. The k representatives provide compact compression over the collection of frequent patterns, making it easier to interpret and use.

Afrati *et al.* (2004) proposed using k itemsets to approximate a collection of frequent itemsets. The measure of approximating a collection of frequent itemsets with k itemsets is defined to be the size of the collection covered by the k itemsets. Yan *et al.* (2005) proposed a profile-based approach to summarize a set of (closed) frequent itemsets into k representatives. A "profile" over a set of similar itemsets is defined as a union of these itemsets, as well as item probability distribution in the supporting transactions. The highlight of profile-based approach is its ability in restoration of individual itemsets and their supports with small error.

Clustering-based compression views frequent patterns as a set of patterns grouped together based on their pattern similarity and frequency support. The condensed pattern-base approach (Pei *et al.* 2002) partitions patterns based on their support and then find the most representative pattern in each group. The representative pattern approach by Xin *et al.* (2005) clusters the set of frequent itemsets based on both pattern similarity and frequency with a tightness measure δ (called δ-cluster). A representative pattern can be selected for each cluster.

Siebes *et al.* (2006) proposed a formulation with the MDL principle – the best set of frequent itemsets is the set that compresses the database best. Heuristic algorithms are developed for finding the subset of frequent itemsets that compresses the database. Since real data is typically subject to noise and measurement error, it is demonstrated through theoretical results that, in the

presence of even low levels of noise, large frequent itemsets are broken into fragments of logarithmic size; thus the itemsets cannot be recovered by a routine application of frequent itemset mining.

Yang *et al.* (2001) proposed two error-tolerant models, termed weak error-tolerant itemsets (ETI) and strong ETI. The support envelope proposed by Steinbach *et al.* (2004) is a tool for exploration and visualization of the high-level structures of association patterns. Asymmetric ETI model is proposed such that the same fraction of errors is allowed in both rows and columns. Seppänen and Mannila (2004) proposed to mine the dense itemsets in the presence of noise where the dense itemsets are the itemsets with a sufficiently large submatrix that exceeds a given density threshold of attributes present. Liu *et al.* (2006) developed a general model forming approximate frequent itemsets (AFI) which controls errors of two directions in matrices formed by transactions and items.

2.6. CHAPTER SUMMARY

This chapter reviewed the various research work conducted in the field of data mining, in particular, regarding frequent pattern mining. Two algorithms, CT-Apriori and CT-PRO are selected for this purpose. The next chapter, Methodology, discusses the functioning of these two algorithms in detail.