Sequential Pattern Mining: A Survey

Qiankun Zhao
Nanyang Technological University, Singapore
and
Sourav S. Bhowmick
Nanyang Technological University, Singapore

1. DATA MINING OVERVIEW

Data mining [Chen et al. 1996] is the process of extracting interesting (non-trivial, implicit, previously unknown and potentially useful) information or patterns from large information repositories such as: relational database, data warehouses, XML repository, etc. Also data mining is known as one of the core processes of Knowledge Discovery in Database (KDD).

Many people take data mining as a synonym for another popular term, Knowledge Discovery in Database (KDD). Alternatively other people treat Data Mining as the core process of KDD. The KDD processes are shown in Figure 1 [Han and Kamber 2000]. Usually there are three processes. One is called preprocessing, which is executed before data mining techniques are applied to the right data. The preprocessing includes data cleaning, integration, selection and transformation. The main process of KDD is the data mining process, in this process different algorithms are applied to produce hidden knowledge. After that comes another process called postprocessing, which evaluates the mining result according to users' requirements and domain knowledge. Regarding the evaluation results, the knowledge can be presented if the result is satisfactory, otherwise we have to run some or all of those processes again until we get the satisfactory result. The actually processes work as follows.

First we need to clean and integrate the databases. Since the data source may come from different databases, which may have some inconsistences and duplications, we must clean the data source by removing those noises or make some compromises. Suppose we have two different databases, different words are used to refer the same thing in their schema. When we try to integrate the two sources we can only choose one of them, if we know that they denote the same thing. And also real world data tend to be incomplete and noisy due to the manual input mistakes. The integrated data sources can be stored in a database, data warehouse or other repositories.

As not all the data in the database are related to our mining task, the second process is to select task related data from the integrated resources and transform them into a format that is ready to be mined. Suppose we want to find which items are often purchased together in a supermarket, while the database that records the purchase history may contains *customer ID*, *items bought*, *transaction time*, *prices*, *number of each items* and so on, but for this specific task we only need *items bought*. After selection of relevant data, the database that we are going to apply our data mining techniques to will be much smaller, consequently the whole process will be

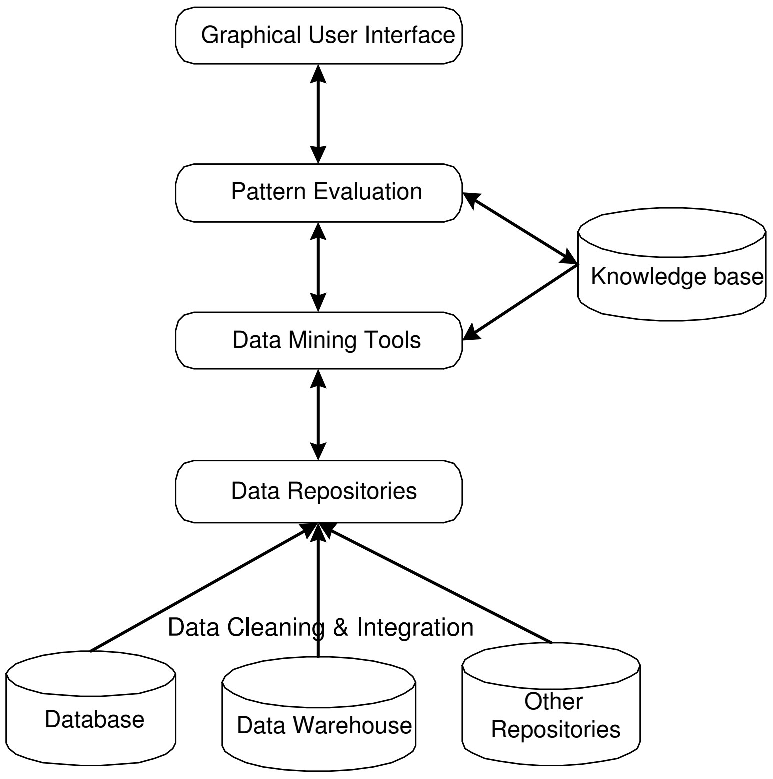

Fig. 1. Knowledge Discovery in Database processes

more efficient.

Various data mining techniques are applied to the data source, different knowledge comes out as the mining result. Those knowledge are evaluated by certain rules, such as the domain knowledge or concepts. After the evaluation, as shown in Figure 1, if the result does not satisfy the requirements or contradicts with the domain knowledge, we have to redo some processes until getting the right results. Depending on the evaluation result we may have to redo the mining or the user may modify his requirements. After we get the knowledge, the final step is to visualize the results. They can be displayed as raw data, tables, decision trees, rules, charts, data cubs or 3D graphics. This process is try to make the data mining results easier to be used and more understandable.

1.1 Types of Mining

Generally speaking, there are two classes of data mining descriptive and prescriptive. Descriptive mining is to summarize or characterize general properties of data in data repository, while prescriptive mining is to perform inference on current data, to make predictions based on the historical data. There are various types of data mining techniques such as association rules, classifications and clustering. Based on those techniques web mining and sequential pattern mining are also well researched. We will review those different types of mining techniques with examples.

Association Rule Association rule mining, one of the most important and well researched techniques of data mining, was first introduced in [Agrawal et al. 1993]. It aims to extract interesting correlations, frequent patterns, associations or casual structures among sets of items in the transaction databases or other data repositories.

Example 1.1. *In an online book store there are always some tips after you purchase some books, for instance, once you bought the book Data Mining Concepts and Techniques, a list of related books such as: Database System 40%, Data Warehouse 25%, will be presented to you as recommendation for further purchasing.*

In the above example, the association rules are: when the book *Data Mining Concepts and Techniques* is brought, 40% of the time the book *Database System* is brought together, and 25% of the time the book *Data Warehouse* is brought together. Those rules discovered from the transaction database of the book store can be used to rearrange the way of how to place those related books, which can further make those rules more strong. Those rules can also be used to help the store to make his market strategies such as: by promotion of the book *Data Mining Concepts and Techniques*, it can blows up the sales of the other two books mentioned in the example. Association rules are widely used in various areas such as telecommunication networks, market and risk management, inventory control etc. Various association mining techniques and algorithms will be briefly introduced and compared later in this Chapter.

Classification Classification [Han and Kamber 2000] is to build (automatically) a model that can classify a class of objects so as to predict the classification or missing attribute value of future objects (whose class may not be known). It is a two-step process. In the first process, based on the collection of *training data set*, a model is constructed to describe the characteristics of a set of data classes or concepts. Since data classes or concepts are predefined, this step is also known as *supervised learning*(i.e., which class the training sample belongs to is provided). In the second step, the model is used to predict the classes of future objects or data.

There are handful techniques for classification [Han and Kamber 2000]. Classification by decision tree was well researched and plenty of algorithms have been designed, Murthy did a comprehensive survey on decision tree induction [Murthy 1998]. Bayesian classification is another technique that can be found in Duda and Hart [Duda and Hart 1973]. Nearest neighbor methods are also discussed in many statistical texts on classification, such as Duda and Hart [Duda and Hart 1973] and James [James 1985]. Many other machine learning and neural network techniques are used to help constructing the classification models.

A typical example of decision tree is shown in Figure 2. A decision tree for the class of *buy_laptop*, indicate whether or not a customer is likely to purchase a laptop. Each internal node represents a decision based on the value of corresponding attribute, also each leaf node represents a class(the value of *buy_laptop*=Yes or No). After this model of *buy_laptop* has been built, we can predict the likelihood of buying laptop based on a new customer's attributes such as *age*, *degree* and

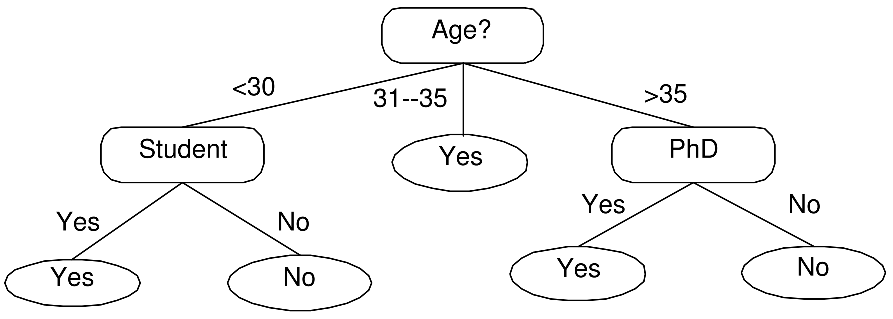

Fig. 2. An example of decision tree

profession. Those information can be used to target customers of certain products or services, especially widely used in insurance and banking.

Clustering As we mentioned before, classification can be taken as supervised learning process, clustering is another mining technique similar to classification. However clustering is a unsupervised learning process. Clustering is the process of grouping a set of physical or abstract objects into classes of similar objects [Han and Kamber 2000], so that objects within the same cluster must be similar to some extend, also they should be dissimilar to those objects in other clusters. In classification which record belongs which class is predefined, while in clustering there is no predefined classes. In clustering, objects are grouped together based on their similarities. Similarity between objects are defined by similarity functions, usually similarities are quantitatively specified as distance or other measures by corresponding domain experts.

Most clustering applications are used in market segmentation. By clustering their customers into different groups, business organizations can provided different personalized services to different group of markets. For example, based on the expense, deposit and draw patterns of the customers, a bank can clustering the market into different groups of people. For different groups of market, the bank can provide different kinds of loans for houses or cars with different budget plans. In this case the bank can provide a better service, and also make sure that all the loans can be reclaimed. A comprehensive survey of current clustering techniques and algorithms is available in [Berkhin 2002].

1.2 Types of Data

Based on the types of data that our mining techniques are applied to, data mining can be classified into different categories.

1.2.1 *Relational database.* Till now most data are stored in relational database, and relational database is one of the biggest resources of our mining objects. As we know relational database is highly structured data repository, data are described by a set of attributes and stored in tables. With the well developed database query languages, data mining on relational database is not difficult. Data mining on relational database mainly focuses on discovering patterns and trends. For example, in the database of a bank, by using some aggregate operators we can find the high spend customers, with data mining techniques we can further classify those customers based on their profile. Also by analyzing the expenses patterns of customers, we can provide certain information to different business organizations. Suppose we found a customer spends 500 dollars every month on fashion, if permitted we can provide this information to the fashion shops.

1.2.2 *Transactional database.* Transactional database refers to the collection of transaction records, in most cases they are sales records. With the popularity of computer and e-commerce, massive transactional databases are available now. Data mining on transactional database focuses on the mining of association rules, finding the correlation between items in the transaction records. An example of association rule on transactional database is show in **Example2.1**. In the later part of association rule mining, most examples are from transactional database, we will elaborate them in detail later.

1.2.3 *Spatial database.* Spatial databases usually contain not only traditional data but also the location or geographic information about the corresponding data. Spatial association rules describe the relationship between one set of features and another set of features in a spatial database, for example {Most business centers in Singapore are *around* City Hall}, the spatial operations that used to describe the correlation can be *within*, *near*, *next to*, etc.. Definitions of spatial association rules and their parameters [Koperski and Han 1995] are identical to those for regular association rules [Agrawal et al. 1993]. The form of spatial association rules is also X⇒Y, where X, Y are sets of predicates and of which some are spatial predicates, and at least one must be a spatial predicate [Koperski and Han 1995; 1996]. Algorithms for mining spatial association rules are similar to association rule mining except consideration of special data, the predicates generation and rule generation processes are based on Apriori, detail of the algorithm for mining spatial association rules were explained in [Koperski and Han 1996]. A spatial association rule mining application GeoMiner [Han et al. 1997] has been developed to extract useful information from a geographical database.

1.2.4 *Temporal and time-series database.* Differ from traditional transaction data, for each temporal data item the corresponding time related attribute is associated. Temporal association rules can be more useful and informative than basic association rules. For example rather than the basic association rule {diapers}⇒{beer}, mining from the temporal data we can get a more insight rule that the support of {diapers}⇒{beer} jumps to 50% during 7pm to 10pm everyday, obviously retailers can make more efficient promotion strategy by using temporal association rule.

In [Han et al. 1999a] and [Mannila et al. 1995] algorithms for mining periodical patterns and episode sequential patterns were introduced respectively. Most of those researches now form a new area of data mining called sequential pattern mining, mining frequent sequential patterns in time series database, which was initiated by Agrawal in [Agrawal and Srikant 1995]. Detail of temporal data mining will be elaborated later in this Chapter.

1.2.5 *World-Wide Web.* As information on the web increases in a phenomena speed and web becomes ubiquitous, most researchers turn to the field of mining web data(Web Mining). Web mining is usually divided into three main categories, web usage mining, web structure mining and web content mining.

Web usage mining concentrates on mining the access patterns of users, so that the structure of the web site can be modified based on the navigation patterns. Different application of mining web logs have been developed to find navigation patterns. Besides improving the web site structure, web usage mining is also valuable for cross marketing strategies, web advertisements and promotion campaigns. Web structure mining focuses in analysis of structures and links in web documents. The basic idea is that those pages that are linked together have some kinds of relationship. With those links, an typical structure mining is to classify those web documents into authoritative pages and hub pages. Authoritative pages are pages that present the original source of information, while hub pages are pages that link to those authoritative pages. Web content includes text, graphic, media etc.. Consequently web content mining includes text mining, multimedia mining and graphic mining.

2. SEQUENTIAL PATTERN MINING

As we know, data are changing all the time, especially data on the web are highly dynamic. As time passes by, new datasets are inserted, old datasets are deleted while some other datasets are updated. It is obvious that time stamp is an important attribute of each dataset, also it is important in the process of data mining and it can give us more accurate and useful information. For example, association rule mining does not take the time stamp into account, the rule can be Buy_A$\Longrightarrow$Buy_B. If we take time stamp into account then we can get more accurate and useful rules such as: Buy_A implies Buy_B within a week, or usually people Buy_A every week. As we can see with the second kind of rules, business organizations can make more accurate and useful prediction and consequently make more sound decisions.

A database consists of sequences of values or events that change with time is called a time-series database [Han and Kamber 2000], a time-series database records the valid time of each dataset. For example, in a time-series database that records the sales transaction of a supermarket, each transaction includes an extra attribute indicate when the transaction happened. Time-series database is widely used to store historical data in a diversity of areas such as, financial data, medical data, scientifical data and so on. Different mining techniques have been designed for mining time-series data [Han and Kamber 2000], basically there are four kinds of patterns we can get from various types of time-series data:

—Trend analysis: trend analysis is to find the evolution patterns of attributes over time, they can be long-term trend movements, cyclic movements or variations, seasonal movements and irregular/random movements. For example, with the history stock prices of Microsoft, we can model the price changes into a function Y=F(t), which can be illustrated as a time series graph. Based on the function we can find the trend of how the price changes, such as every Monday the price will increase by 4% and every Friday the price will drop by around 5%. This method is widely used in the analysis of stock market.

—Similarity search: similarity search tries to find sequences that differ only slightly. Similarity searching is a blurry matching process that can tolerate some differences within a certain threshold. Based on the length of sequences we are trying to match, sequence matching can be classified as: subsequence matching and whole sequence matching. Suppose we transform the data of stock prices into curves, those curves include many different shapes such as *up, sharp up, down, big down.* The basic similarity search example is based on the curve and shapes of one stock to find other stocks that have the similar curves and shapes. There exists many different kinds of similarity measurements. One of the classic measurement of similarity is Euclidean distance.

—Sequential patterns: sequential pattern mining is trying to find the relationships between occurrences of sequential events, to find if there exist any specific order of the occurrences. We can find the sequential patterns of specific individual items, also we can find the sequential patterns cross different items. Sequential pattern mining is widely used in analyzing of DNA sequence. An example of sequential patterns is that every time Microsoft stock drops 5%, IBM stock will also drops at least 4% within three days.

—Periodical patterns: periodical patterns are those recurring patterns in the time-series database, periodicity can be daily, weekly, monthly, seasonal or yearly. Obviously, periodical patterns mining can be viewed as sequential pattern mining by taking the periodical sequences as a set of sequences. In the customer transaction database, each sequence represents a sequence of items that certain customer bought. For periodical pattern mining, we have a sequence of data for a very long time, but we can partition them based on periods and get a set of sequences. An example of periodical pattern is that if a restaurant received many tea customers during 3:00-5:00 pm, dinner will sell well between 8:00-9:00 pm.

Since last decade time-related data mining has been a hot research area. However, sequential database consists of sequences of ordered events with or without concrete notion of time. Sequential database is a special case of time series database, consequently most researches in sequential pattern mining focus on two main issues. The first issue is sequential pattern mining, aiming at finding the frequently occurred sequences to describe the data or predict future data. The second issue is periodical pattern mining, which can be viewed as sequential pattern mining. In this section we will introduce different techniques of sequential pattern mining.

2.1 Sequential Patten Mining Problem

Example 2.1. *From a book store's transaction database history, we can find the frequent sequential purchasing patterns, for example 80% customers who brought the book Database Management typically bought the book Data Warehouse and then brought the book Web Information System with certain time gap.*

Sequential pattern mining was first introduced in [Agrawal and Srikant 1995]. In the above example, all those books need not to be brought at the same time or consecutively, the most important thing is the order in which those books are brought and they are bought by the same customer. 80% here represents the percentage of customers who comply this purchasing habit. Sequential pattern can be widely used in different areas, such as mining user access patterns for the web sites, using the history of symptoms to predict certain kind of disease, also by using sequential pattern mining, the retailers can make their inventory control more efficient.

Sequential patterns indicate the correlation between transactions while association rule represents intra transaction relationships. In association rule mining of transaction database, the mining results are about which items are brought together frequently, those items must come from the same transaction. While the result of sequential pattern mining are about which items are brought in a certain order by the same customer, those items come from different transactions.

Let D be a database of customer transactions, I=I_1, I_2, $\cdots$, I_m be a set of m distinct attributes called items, T be transaction that includes {customer-id, transaction-time, item-purchased }, s_i be an itemsets, which contains a set of items from *I*, S be a sequence that consists of an ordered list of itemsets $\langle$ s_1, s_2, $\cdots$, s_n $\rangle$. No customer has more than one transaction with the same transaction-time. As shown in Table I, 10, 20, 30 $\cdots$ are items, Table I(a) is a list of transactions, (10, 20) $\cdots$ are itemsets, while $\langle(30)(90)\rangle \cdots$ are sequences. All the items in the same round bracket have the same transaction time, while there is an order between round brackets within the large angle bracket.

Sequential pattern is a sequence of itemsets that frequently occurred in a specific order, all items in the same itemsets are supposed to have the same transaction-time value or within a time gap. Usually all the transactions of a customer are together viewed as a sequence, usually called customer-sequence, where each transaction is represented as an itemsets in that sequence, all the transactions are list in a certain order with regard to the transaction-time.

Contain, a sequence $\langle a_1, a_2, \cdots, a_n\rangle$ is contained in another sequence $\langle b_1, b_2, \cdots, b_m\rangle$, if $i_1 < i_2 < \cdots < i_n$ such that $a_1 \subseteq b_{i1}, a2 \subseteq b_{i2}, \cdots a_n \subseteq b_{in}$. For example, the sequence $\langle(3)(6, 7, 9)(7, 9)\rangle$ is contained in $\langle(2)(3)(6, 7, 8, 9)(7)(7, 9)\rangle$, since (3)⊆(3),(6,7,9)⊆(6,7,8,9), (7,9)⊆(7,9). However, sequence $\langle(2)(3)\rangle$ is not contained in sequence $\langle(2, 3)\rangle$, since the former sequence means 3 is bought after 2 being bought, while the latter represents item 2 and 3 being bought together. A sequence is *maximal* if it is not contained in any other sequences.

Support, a customer support a sequence s if s is contained in the corresponding customer-sequence, the support of sequence s is defined as the fraction of customers who support this sequence.

$$Support(s) = \frac{Number\ of\ support\ customers}{Total\ number\ of\ customers}$$

Sequential pattern mining is the process of extracting certain sequential patterns whose support exceed a predefined minimal support threshold. Since the number of sequences can be very large, and users have different interests and requirements, to get the most interesting sequential patterns, usually a minimum support is pre-defined by users. By using the minimum support we can prune out those sequential patterns of no interest, consequently make the mining process more efficient. Obviously a higher support of sequential pattern is desired for more useful and interesting sequential patterns. However some sequential patterns that do not satisfy the support threshold are still interesting. Another metric called *surprise* has been introduced to measure the interestingness of those sequences in [Yang

Customer-id	Transaction-time	Purchased-items
1	Oct 23' 02	30
1	Oct 28' 02	90
2	Oct 18' 02	10, 20
2	Oct 21' 02	30
2	Oct 27' 02	40, 60, 70
3	Oct 15' 02	30, 50, 70
4	Oct 08' 02	30
4	Oct 16' 02	40, 70
4	Oct 25' 02	90
5	Oct 20' 02	90

(a) Sorted Transaction Data

Large Itemsets	Mapped To
(30)	1
(40)	2
(70)	3
(40, 70)	4
(90)	5

(b) Large Itemsets MinSupp=40%

Customer-id	Customer Sequence	Transformed DB	After Mapping
1	⟨(30)(90)⟩	⟨{(30)}{(90)}⟩	⟨{1}{5}⟩
2	⟨(10, 20)(30)(40, 60,70)⟩	⟨{(30)}{(40)(70) (40,70)}⟩	⟨{1}{2,3,4}⟩
3	⟨(30, 50, 70)⟩	⟨{(30)(70)}⟩	⟨{1, 3}⟩
4	⟨(30)(40, 70)(90)⟩	⟨{(30)}{(40)(70)(40, 70)}{(90)}⟩	⟨{1}{2, 3, 4}{5}⟩
5	⟨(90)⟩	⟨{(90)}⟩	⟨{5}⟩

(c) Transformed Database

Table I. Sequential Database

et al. 2001], a sequence s is surprising pattern if its occurrence differs greatly from the expected occurrence by treading every items equal. In the *surprise* metric the information gain in information theory was proposed to measure the overall degree of *surprise*, details are available in [Yang et al. 2001].

Sequential pattern mining is used in a great spectrum of areas. In computational biology, sequential pattern mining is used to analyze the mutation patterns of different amino acids. Business organizations use sequential pattern mining to study customer behaviors. Sequential pattern mining is also used in system performance analysis and telecommunication network analysis.

2.2 Sequential Pattern Mining Approaches

Sequential pattern mining has been intensively studied during recent years, there exists a great diversity of algorithms for sequential pattern mining. In this Section we first introduce some general and basic algorithms for sequential pattern mining, extensions of those algorithms for special purposes, such as multi-dimensional sequential pattern mining and incremental mining are covered later on. Also periodical pattern mining is elaborated as an extension of sequential pattern mining.

2.2.1 *General Algorithms for Sequential Pattern Mining.* Most of the basic and earlier algorithms for sequential pattern mining are based on the Apriori property proposed in association rule mining [Agrawal and Srikant 1994], the property states that any sub-pattern of a frequent pattern must be frequent. Based on this heuristic, a series of Apriori-like algorithms have been proposed: AprioriAll, AprioriSome, DynamicSome in [Agrawal and Srikant 1995] , GSP [Srikant and Agrawal 1996] and SPADE [Zaki 2001]. Later on another a series of data projection based algorithms were proposed, which includes FreeSpan [Han et al. 2000] and PrefixSpan [Pei et al. 2001]. SPADE [Zaki 2001] is a lattice based algorithm, MEMISP [Lin and Lee 2002] is a memory indexing based approach , while SPIRIT [Garofalakis et al. 1999] integrates constraints by using regular expression. Most of those algorithms will be briefly explained in this subsection.

In order to make it easier for us to compare those algorithms we use the same sequential database, a sequential database from a supermarket, to explain how those algorithms work. This sequential data records the purchasing history of its customers over certain period of time. Suppose during the preprocess all those attributes that are not relevant or useful to our mining task are pruned, only those useful attributes are left as shown in Table I(a).

2.2.2 *AprioriAll.* Sequential pattern mining was first introduced in [Agrawal and Srikant 1995] by Agrawal, three Apriori based algorithms were proposed. Given the transaction database with three attributes *customer-id*, *transaction-time* and *purchased-items*, the mining process were decomposed into five phases(Example of the first three phases are shown in Table I):

—**Sort Phase**: the original transaction database is sorted with customer-id as the major key and transaction time as the minor key, the result is set of customer sequences. Table I(a) shows the sorted transaction data.

—**L-itemsets Phase:** the sorted database is scanned to obtain large 1-itemsets according to the predefined support threshold. Suppose the minimal support is 40%, in this case the minimal support count is 2, the result of large 1-itemsets is listed in Table I(b).

—**Transformation Phase:** the customer sequences are replaced by those large itemsets they contain, all the large itemsets are mapped into a series of integers to make the mining more efficient. At the end of this phase the original database is transformed into set of customer sequences represented by those large itemsets. For example transactions with customer-id 1 are transformed into customer sequence$\langle(30)(90)\rangle$, this transaction contains two large 1-itemsets (30) and (90) as shown in the Transformed DB, while they are mapped to $\langle(1)(5)\rangle$ according to the map table in Table I(b). Finally the result database is shown in Table I(c).

—**Sequence Phase:** all frequent sequential patterns are generated from the transformed sequential database.

—**Maximal Phase:** those sequential patterns that are contained in other super sequential patterns are pruned in this phase, since we are only interested in maximum sequential patterns.

Since most of the phases are straightforward, researches focused on the sequence phase in [Agrawal and Srikant 1995]. AprioriAll was proposed to find the frequent sequential patterns, which is stated as the fourth phase.

AprioriAll is based on the Apriori algorithm in association rule mining, similarly there are two subprocess. The first is to generate those sequences that may be frequent, which is also called candidate sequences. Then the sequential database is scanned to check the support of each candidate to determine frequent sequential patterns according to minimal support. Since the time cost of the second process is determined by the number of passes over the database and number of candidates, most researchers mainly concern about the candidate generation process and the passes over the database.

The candidate generation process is similar to the AprioriGen in [Agrawal and Srikant 1994]. The Apriori property is also used to prune those candidate sequences whose subsequence is not frequent. The difference is that when we generate the candidate by joining the frequent patterns in the previous pass, different order of combination make different candidates. For example: from $\{1\}$, $\{2\}$ we can generate two candidates $\langle\{1\}\ \{2\}\ \rangle$ and $\langle\{2\}\ \{1\}\rangle$, but in association rule mining we only generate $\{1, 2\}$. Obviously the number of candidate sequences double the size of the candidate itemsets in association rule mining during the generation of 2-candidate sequences. Table II shows how to generate candidate 4-sequences by joining large 3-sequences. By scanning the large 3-itemsets, we find that the first itemsets $\langle$ 1, 2, 3 $\rangle$ and second itemsets $\langle$ 1, 2, 4 $\rangle$ share their first two items, according to the join condition of Apriori they are joined to produce the two candidates $\langle$ 1, 2, 3, 4 $\rangle$ $\langle$ 1, 2, 4, 3 $\rangle$. Similarly other candidate 4-sequences are generated.

Large 3-sequences	Candidate 4-sequences
⟨ 1, 2, 3 ⟩	⟨ 1, 2, 3, 4 ⟩
⟨ 1, 2, 4 ⟩	⟨ 1, 2, 4, 3 ⟩
⟨ 1, 3, 4 ⟩	⟨ 1, 2, 3, 5 ⟩
⟨ 1, 3, 5 ⟩	⟨ 1, 2, 5, 3 ⟩
⟨ 2, 3, 4 ⟩	

Table II. AprioriAll Candidate Generation L_3 to C_4

The check process is quite straight forward, by scanning the database the support counts of those candidate sequences can be obtained. By comparing them with the support threshold we can get those frequent sequential patterns. In the maximal sequence phase those sequences whose super-sequences are frequent are pruned out, since we are interested only in maximal sequences.

AprioriAll was the first algorithm for sequential pattern mining, it is based on the naive approach of Apriori association rule mining. The main drawback of AprioriAll is that too many passes over the database is required and too many candidates are generated. Based on Apriori algorithm for association rule mining, AprioriAll is not so efficient but it is the basis of many efficient algorithm developed later.

2.2.3 *GSP(Generalized Sequential Pattern).* GSP [Srikant and Agrawal 1996] is also an Apriori based algorithm for sequential pattern mining, however it integrates with time constraints and relaxes the definition of transaction, also it considers the knowledge of taxonomies. For time constraints, *maximum gap* and *minimal gap* are defined to specified the gap between any two adjacent transactions in the sequence. If the distance is not in the range between *maximum gap* and *minimal gap* then this two can not be taken as two consecutive transactions in a sequence. This algorithm relaxes the definition of transaction by using a sliding window, which means if the distance between the maximal transaction time and the minimal transaction time of those items is no bigger than the sliding window those items can be taken as in the same transaction. The taxonomies is applied to generate multiple level sequential patterns. With those new factors sequential pattern mining can be defined as [Srikant and Agrawal 1996]: given a sequence data D, a taxonomy T, user-defined *min-gap* and *max-gap* time constraints, a user-defined sliding *window-size*, to find all sequences whose *support* is greater than the user-defined *minimum support.*

Like other algorithms GSP has two subprocess: candidate pattern generation and frequent pattern generation.

In the candidate generation process, similar to the AprioriGen function in association rule mining [Agrawal et al. 1993] candidate k-sequences are generated based on the large (k-1)-sequences. Given a sequence $s = \langle s_1, s_2, \cdots, s_n \rangle$ and subsequence c, c is a contiguous subsequence of s if any of the following conditions hold: 1). c is derived from s by dropping an item from either s_1 or s_n. 2). c is derived from s by dropping an item from an element s_j that has at least 2 items. 3). c is a contiguous subsequence of $\dot{c}$, and $\dot{c}$ is a contiguous subsequence of s. The candidate sequences are generated in two steps as shown in Table III. Join Phase, candidate k-sequences are generate by joining two (k-1)-sequences that have the same contiguous subsequences, when we joining the two sequences the item can be inserted as a part of the element or as a separated element. For example, $\langle(1,2)(3)\rangle$ and $\langle(1,2)(4)\rangle$ have the same contiguous subsequence $\langle(1,2)\rangle$, based on those candidate 4-sequence $\langle(1,2)(3,4)\rangle$, $\langle(1,2)(3)(4)\rangle$ and $\langle(1,2)(4)(3)\rangle$ can be generated. Prune Phase, those candidate sequences that have a contiguous subsequence whose support count is less than the minimal support are deleted. Also it uses the hash-tree structure [Park et al. 1995] to reduce the number of candidates to be checked in the next phase.

In AprioriAll it is easy to get the support counts of all those candidate sequences since all the sequences in the database are represented by sub-sequences they contain. With the introduction of maximal and minimal gaps, in GSP it is difficult to get the support counts of candidate sequences. The contain test was introduced with two phases: forward phase and backward phase, which are repeated until all the elements are found. Following is the process of checking if the data-sequence d contain a candidate sequence s :

—Forward Phase: we find successive elements of s in d as long as the difference between the end-time of the element and the start-time of the previous element

Large 3-sequences	Candidate 4-sequences after join	Candidate 4-sequences after pruning
⟨ (1, 2) (3) ⟩	⟨ (1, 2) (3, 4) ⟩	⟨ (1, 2) (3, 4) ⟩
⟨ (1, 2) (4) ⟩	⟨ (1, 2) (3, 5) ⟩	
⟨ (1) (3, 4) ⟩	⟨ (1, 2) (3)(4) ⟩	
⟨ (1, 3) (5) ⟩	⟨ (2)(3, 4)(5) ⟩	
⟨ (2) (3, 4) ⟩	⟨ (1, 2) (4)(3) ⟩	
⟨ (2) (3) (5) ⟩		

Table III. GSP Candidate Generation L_3 to C_4

is less than max-gap, if the difference is more than max-gap, we switch to the backward phase. If an element is not found then sequence s is not contained in d.

—Backward Phase: we try to pull up the previous element, suppose s_i is the current element and end-time(s_i)=t, we check if there exist transactions containing s_{i-1} and their transaction-times are after *t-max-gap*. Since after pulling up s_{i-1}, the difference between s_{i-1} and s_{i-2} may not satisfy the gap constraints, the backward pulls back until the difference of s_{i-1} and s_{i-2} satisfies the max-gap or the first element has been pulled up. Then the algorithm switches to the forward phase, if any element can not be pulled up the data-sequence d does not contain s.

The taxonomies are applied to the sequence by extending sequences with corresponding taxonomies, sequences are replaced with their extended sequences. The number of rules becomes large when extended sequences are mined because the sequences become more dense, consequently there are many redundant rules. To avoid uninteresting sequences, first the ancestors are pre-computed for each item, ancestors that are not in the candidate sequences are dropped. Secondly they do not count sequential patterns that contain both the item and its ancestors.

For the Apriori based algorithms, the number of candidate sequences is quite large and they require many passes over the whole database as well. But the two approaches are the basis of further researches on mining sequential pattern, at least all the sequential patterns can be generated though they are not so efficient. GSP performs relatively better than AprioriAll, in GSP the number of candidate sequences is much smaller, also time constraints, taxonomies are integrated during the sequential patterns mining process to produce more knowledge.

Later A GSP-based algorithm called MFS(Mining Frequent Sequences) [Zhang et al. 2001], rather than scan the database many times as the length of the maximal sequence, MFS proposed a two-stage algorithm. First a sample of the database is mined to get the rough estimation of frequent sequences, based on those estimation the database is scanned to check and refine the candidate sequences until no more frequent sequences can be found. The difference between MFS and GSP is the function of candidate generation: in GSP each time only those candidates of the same length are generated by joining those frequent sequences in the previous pass, but in MFS candidates of different lengthes are generated by joining all those known frequent sequences. Experiments shown that MFS generates the same set of frequent sequences as GSP while it outperforms GSP by reducing the I/O cost.

A universal formulation of sequential patterns is introduced in [Joshi et al. 2001]. Different kinds constraints such as structure, time, item are discussed to be integrated into the universal system, also corresponding counting methods of sequential patterns and how to set thresholds are explained. A GSP based algorithm is introduced with different count methods to explain how to use the universal formulation of sequential patterns.

2.2.4 *PrefixSpan.* PrefixSpan [Pei et al. 2001] is a more efficient algorithm for mining sequential patterns comparing with GSP and Apriori. PrefixSpan is also capable of dealing very large database. PrefixSpan mainly employs the method of database projection to make the database for next pass much smaller and consequently make the algorithm more speedy, also in PrefixSpan there is no need for candidates generation only recursively project the database according to their prefix. Different projection methods were discussed for PrefixSpan: level-by-level projection, bi-level projection and pseudo projection.

The process of PrefixSpan is quite different from the two algorithms mentioned above. Since the item order within an element does not affect the sequential mining, we suppose that items within elements are in alphabetical order. The first step of PrefixSpan is to scan the sequential database to get the length-1 sequence, which is in fact the large 1-itemsets. Then the sequential database is divided into different partitions according the number of length-1 sequence, each partition is the projection of the sequential database that take the corresponding length-1 sequences as prefix. For example, with the frequent 1-sequence $\langle(30)\rangle$ as the prefix, the projected database is shown in Table IV(b). The projected databases only contain the postfix of these sequences, by scanning the projected database all the length-2 sequential patterns that have the parent length-1 sequential patterns as prefix can be generated. Then the projected database is partitioned again by those length-2 sequential patterns. The same process are executed recursively until the projected database is empty or no more frequent length-k sequential patterns can be generated.

The method mentioned above is called level-by-level projection, there is no candidate generation process, the main cost of this method is the time and space used to construct projected databases as shown in Table IV(b). Another projection method called bi-level projection is proposed to reduce the number and size of projected databases. The first step is the same, by scanning the sequential database we can get the frequent 1-sequence, in the second step instead of constructing projected database a n× n triangle matrix M is constructed as shown in Table IV (d). The matrix represents all the supports of length-2 sequences, for example M[$\langle(40)\rangle$, $\langle(30)\rangle$]=(0, 2, 1) means supports of $\langle(40)(30)\rangle$, $\langle(30)(40)\rangle$ and $\langle(40, 30)\rangle$ are 0, 2 and 1 respectively. After that the S-matrix projected databases are constructed for those frequent length-2 sequences, all the processes iterated until the projected database becomes empty or no frequent sequence can be found. By using the triangle S-matrix to represent all supports of length-2 sequences, the number of projected databases becomes smaller and the requires less space.

Pseudo projection is another method designed to make the projection more ef-

Customer-id	Customer Sequence
1	⟨(30)(90)⟩
2	⟨(10, 20)(30)(40, 60,70)⟩
3	⟨(30, 50, 70)⟩
4	⟨(30)(40, 70)(90)⟩
5	⟨(90)⟩

(a) Transformed Database

Large Itemsets	Projected Database
⟨(30)⟩	⟨(90)⟩ ⟨(40, 60,70)⟩ ⟨(_, 50, 70)⟩ ⟨(40, 70)(90)⟩
⟨(40)⟩	⟨(_, 60,70)⟩ ⟨(_, 70)(90)⟩
⟨(70)⟩	⟨(90)⟩
⟨(40, 70)⟩	⟨(90)⟩
⟨(90)⟩	

(b) Level-by-Level Projected Database

Large Itemsets
⟨(30)⟩
⟨(40)⟩
⟨(70)⟩
⟨(40, 70)⟩
⟨(90)⟩

(c) Large Itemsets

⟨(30)⟩	0				
⟨(40)⟩	(0, 2, 0)	0			
⟨(70)⟩	(0, 2, 1)	(0, 0, 2)	0		
⟨(40, 70)⟩	(0, 2, 0)	(0, 0, 2)	(0, 0, 2)	0	
⟨(90)⟩	(0, 2, 0)	(0, 1, 0)	(0, 1, 0)	(0, 1, 0)	0
	⟨(30)⟩	⟨(40)⟩	⟨(70)⟩	⟨(40, 70)⟩	⟨(90)⟩

(d) The S-matrix

Table IV. Sequential Database

ficient when the projected database can be fitted in main memory. Actually no physical projection database is constructed, each postfix is represent by a pair of pointer and offset value. Avoided copying the database, pseudo projection is more efficient than the other two projection methods, however the limitation is that the size of the database must can be fitted into the main memory.

PrefixSpan mainly avoids generating and counting candidate sequences, which is the most time-consuming part of AprioriAll and GSP. By using projection, the database PrefixSpan scans every time is much smaller than the original database. The main cost of PrefixSpan is the projected database generation process, in order to improve the performance a bi-level projection method that uses the triangle S-Matrix is introduced. Also if the sequential pattern is very long, GSP and AprioriAll are infeasible and inefficient, while PrefixSpan can handle this more efficiently.

2.2.5 *SPADE.* SPADE [Zaki 2001] is an algorithm proposed to find frequent sequences using efficient lattice search techniques and simple joins. All the sequences are discovered with only three passes over the database, it also decomposes the mining problem into smaller subproblems, which can be fitted in the main memory. SPADE outperforms AprioriAll and GSP by a factor of two through experiments.

In this approach, the sequential database is transformed into a vertical id-list database format, in which each id is associated with corresponding items and the time stamp. The vertical database of Table I(a) is shown in Table V(a). For item 30, there support count is 4, it occurs with SID 1, 2, 3 and 4 at TID 23, 21, 15 and 8 respectively. By scanning the vertical database, frequent 1-sequences can be generated with the minimum support. For the 2-sequences, the original database

(30)		(90)		(70)		(40)	
SID	TID	SID	TID	SID	TID	SID	TID
1	23	1	28	2	27	2	27
2	21	4	25	3	15	4	16
3	15	5	20	4	16		
4	8						

(a) Vertical Id-List

SID	(Item, TID) pairs
1	(*30*, 23)(*90*, 28)
2	(*30*, 21)(*40*, 27)(*70*,27)
3	(*30*, 15)(*70*, 15)
4	(*30*, 8)(*40*, 16)(*70*, 16)
51	(*90*, 20)

(b) Vertical to horizontal database

Table V. SPADE Process

is scanned again and the new vertical to horizontal database is created by grouping those items with SID and in increase order of TID. The result vertical to horizontal database is shown in Table V(b). By scanning the vertical to horizontal database 2-sequences are generated. All the 2-item sequence found are used to construct the lattice, which is quite large to be fitted in main memory. However the lattice can be further decomposed to different classes, sequences that have the same prefix items belong to the same class. By decomposition the lattice is partitioned into small partitions that can be fitted in main memory. During the third scanning of the database all those longer sequences are enumerated by using *temporal join* [Zaki 2001].

There are two methods of enumerating frequent sequences of a class: Breadth-First Search (BFS) and Depth-First Search (DFS). In BFS the classes are generated in a recursive bottom-up manner. For example to generate the 3-item sequences all the 2-item sequences have to be processed, but in DFS only one 2-item sequence and a k-item sequence are further needed to generate (k+1)-item sequence. BFS need much bigger main memory to store all the consecutive 2-item sequences, but DFS just need to store the last 2-item sequence of the newly generated k-item sequences. However BFS has more information to prune the candidate k-item sequences, and the possibility of being large of those sequences generated by BFS is much bigger than those generated by DFS. All the k-item patterns are discovered by temporal or equality joining the frequent (k-1)-item patterns which have the same (k-2)-item prefix, while there are three possible joining results as candidate generation process in GSP [Srikant and Agrawal 1996], the Apriori property pruning technique is also employed in SPADE.

2.2.6 *MEMISP.* All those aforementioned sequential pattern mining algorithms either require many passes over the databases as in GSP, or generate many intermediate projected databases as in PrefixSpan. Another approach called MEMory Indexing for Sequential Pattern mining(MEMISP) is introduced in [Lin and Lee 2002]. MEMISP only requires one pass over the database, at most two passes for very large database, and it avoids generation of candidates and projection of intermediate database as well.

In this approach, MEMISP uses a recursive searching and indexing strategy to generate all the sequential patterns from the data sequences stored in memory. Some terms are defined in this algorithm. Given a pattern ρ and a frequent item x in the sequence database, ρ' is a type-1 pattern if it can be formed by appending the itemsets(x) as a new element to ρ, ρ' is a type-2 pattern if it can be formed by extending the last element of ρ with itemsets(x). The frequent itemset x is called a *stem*, while the sequential pattern ρ is the **P**refix **pat**tern (P-pat) of ρ'. The MEMISP algorithm works as follows.

The first phase is to scan the database and write it into memory to form the MDB(Memory Database). During this process the support counts of those length 1 sequences are recorded to get the frequent 1-sequences. Those frequent 1-sequences will be used as **stem** of **type-1** pattern with respect to **P-pat**=$\langle\ \rangle$. The second phase is to output the sequential pattern ρ formed by current P-pat and stem x and construct the index set ρ-idx. Index set ρ-idx is the collection of these(ptr_ds, pos) pairs. A (ptr_ds, *pos*) pair for each data sequence *ds* in MDB is allocated, if *ds* contains x, where prt_ds is a pointer to *ds* and *pos* is the first occurring position of x in *ds*. The third phase is using index set ρ-idx and MDB to find stems with respect to P-pat=ρ. To find any sequential patterns that having current pattern ρ as its P-pat, first they find those ρ-idx pairs whose ptr_ps points to the data sequences that contain ρ, those items that occur after the corresponding *pos* positions are taken as potential *stem*. The count of these items increase by one every time when they appear after the *pos* position, with the support count the *stem* can be determined frequent or not. Then we turn to the second phase. The second and third phases are executed recursively until no further *stem* can be found. During the whole process no further scan of the database is needed, the index is recursively updated. This consequently make MEMISP more efficient than others.

With more and more memory installed in the computer, most databases can be easily fitted into the main memory. However some very large databases still can not resident in the main memory. For those databases, they are partitioned into small ones that can be stored in memory and apply MEMISP to each of them to get the sequential patterns, the candidate sequential patterns of the whole database are the collection of patterns outputted from each partitions. To determine final frequent sequential patterns another scanning of the whole database is needed to check the actual support, only two passes over the whole database is needed for those large databases. Experiment results showed that MEMISP is more efficient than GSP and PrefixSpan, it also has a good linear scalability to the size of database and the number of data sequences.

2.2.7 *SPIRIT(Sequential Pattern mIning with Regular expressIon consTraints).* Many different algorithms for sequential pattern mining in time series database have been designed and implemented, most of those algorithms aims to improve the efficiency of the ad-hoc algorithms. In reality, users are interested in different sequential patterns even for the same database, users are interested in some specific patterns rather than the whole possible sequential patterns. SPIRIT [Garofalakis et al. 1999] is a method of mining user-specified sequential patterns by using regular expression constraints. This method avoid wastage of computing effort for mining patterns that users are not interested in, it also avoids overwhelming users with potentially useless patterns.

In this approach, user's specific requirements are stated in regular expression, denoted by C, four algorithms of SPIRIT are introduced. They proposed a regular expression and a series of operators that can read and interpret the regular expression of constraints. SPIRIT(N) is the most naive approach with regular expression constraints, it uses only the including constraints to pruning the candidate sequences, other processes is the same as in GSP. However in SPIRIT(L) and SPIRIT(V), every candidate k-sequence are checked according to the regular expression by using different operator such as legal and valid. With those operators the pruning techniques become more efficient. SPIRIT(R) is totally different in the candidate generation process, rather than join the sequences of less lengthes the candidate are generated by enumerate the possible combination of path traverse of the regular expression. Detail and example of the four algorithms are available in the original paper.

The difference between those four algorithms is the extend to which the constraints are pushed into the candidate generation and pruning processes. Experiments have been conducted with synthetic and real-life data sets, the results showed that when the constraints are highly selective SPIRIT(R) outperforms the other three algorithms, while in most cases SPIRIT(V) is the overall winner over the entire range of regular expression constraints. SPIRIT(N) is the most straightforward and heuristic method since it enforce only limited constraint to the mining process.

2.2.8 *Multi-Dimensional Sequential Pattern Mining.* Besides mining sequential patterns in a single dimension, which means while finding the frequent patterns we only consider one attribute together with time stamps, mining multiple dimensional sequential patterns can give us more informative and useful patterns. For example we may get the frequent pattern from the supermarket database that most people who buy package 1 also buy package 2 in a defined time interval by employing general sequential pattern mining. However using multiple dimensional sequential pattern mining we can further find different groups of people have different purchase patterns. For example, students always buy A within a week after they buy B, while those sequential rules do not hold for other groups. Multi-dimensional sequential pattern mining is first introduced in [Pinto et al. 2001]. In multi-dimensional sequential pattern mining, different attributes of the transaction ID were introduced and formed a multi-dimensional sequential data sets as shown in Table VI. The aim of this special mining is to get more interesting sequential patterns with different dimensional attributes. One direct extension of PrefixSpan [Pei et al. 2001] algorithm and two approaches combined with PrefixSpan and BUC(Bottom Up Computation)-like [Beyer and Ramakrishnan 1999] algorithms will be briefly introduced.

cid	cust-grp	city	age-grp	sequence
10	business	Boston	middle	⟨(b,d)cba⟩
20	professional	Chicago	young	⟨(b,f)(c, e)(f, g)⟩
30	business	Chicago	middle	⟨(a,h)abf⟩
40	education	New York	retired	⟨(b,e)(c, e)⟩

Table VI. A Multi-dimensional sequence database

The first approach is called UNISEQ(Unique Sequence), in this approach the multi-dimensional attributes are embedded in the sequential database by adding a new element in the sequence database. With the newly formed sequential database in Table VII, PrefixSpan is employed in this method. The PrefixSpan finds all the single-item large sequences during the first scan by counting the occurrences. For every single-item frequent sequence the database containing the prefix sequence is projected and the postfix sequences of them are obtained, further we get the frequent single-item sequence in the projected database and use the two single-items as a prefix to another iteration. The iteration stops when there are no further prefix in the projected database, detail of PrefixSpan has been explained in the last section.

cid	MD-extension of sequence
10	⟨(business, Boston, middle) (b,d)cba⟩
20	⟨ (professional, Chicago, young)(b,f)(c, e)(f, g)⟩
30	⟨ (business, Chicago, middle) (a,h)abf⟩
40	⟨ (education, New York, retired)(b,e)(c, e)⟩

Table VII. A MD-extension database

Another approach is to partition the combined sequence database into two, one part is the dimensional information another is the former sequences. The problem of finding multiple dimensional sequential patterns can be decomposed into two sub-problem: mining multi-dimensional patterns and mining sequential patterns. BUC-like algorithm is used to mining multi-dimensional patterns, first the database is sorted in alphabetical order with regard of values in dimension. The frequent one dimensional items can be obtained after the first scan of the database. With those tuples that contain this frequent dimensional items, similar method is used to find the frequent dimensional two-itemsets pattern and so on until there is not further longer dimensional frequent patterns. All the processes are carried out for each dimensional item and finally all the frequent dimensional itemsets patterns are obtained. The original sequence database is projected to construct a new sequential database with those tuples that contain those frequent dimensional itemsets. PrefixSpan is used to mining sequential patterns in the newly created database. Actually there are two combinations considering which part should we mine first. For Dim-Seq, MD(Multiple Dimension)-patterns are found first and for each MD-pattern the projected database is formed and sequential pattern mining is applied to those projected database. For Seq-Dim, sequential patterns are mined first, for those sequential patterns the projected database were formed and MD-pattern mining are applied to each projected database.

Experiments have been conducted to evaluate the performance of the three algorithms: UniSeq, Dim-Seq and Seq-Dim. The results showed that Seq-Dim outperforms the other two in most cases. Since usually the MD-patterns are very shorter comparing with the sequential patterns, after mining the sequential patterns the projected database contains only those tuples with frequent sequential patterns, thus the MD-pattern mining is more efficient and fruitful. When the dimensionality is low, UniSeq wins because the BUC-like mining gets little advantage when the multi-dimensional pattern is quite short, the main factor of UniSeq is the cost of mining elements with the multi-dimensional values. Dim-Seq is not so efficient comparing with the other two methods, since many multi-dimensional patterns may not lead to multi-dimensional sequential patterns, there exists many wastage of efforts during this sequential pattern mining process.

Till now multi-dimensional sequential pattern mining has not been explored together with other constraints such as time gap, sliding window and item constraints. The future research issues in multi-dimensional sequential pattern mining can be: multi-dimensional sequential pattern mining with constraints, interactive multi-dimensional sequential pattern mining or multi-dimensional sequential pattern mining integrated with taxonomies and hierarchies.

2.2.9 *Incremental Mining of Sequential Patterns.* Incremental mining always involves in all data mining techniques, since all the databases are updated over time. It is more important in sequential patten mining since our mining sequential data is always changing over time. For example the sequential data of weather history is updated everyday with a new element. Although all the sequential pattern mining algorithms can be applied into sequential pattern mining by re-mining the whole database whenever it changes, researchers are trying to find some algorithms that are more efficient in dealing with incremental mining. In [Zhang et al. 2002] a GSP based and a MFS based incremental mining algorithms are introduced, while in [Parthasarathy et al. 1999] a SPADE based incremental mining is introduced, also ISE and IUS are introduced in [MASSEGLIA et al. 2000] and [Zheng et al.] respectively. The problem of when to update the sequential patterns is discussed in [Zheng et al. 2002].

In [Zhang et al. 2002] the incremental mining problem is defined as following : Given a database D of sequences, it is updated by inserting a set of sequences $\triangle^+$ and deleting a set of sequences $\triangle^-$, the updated database is denoted as D', the set of unchanged sequences are denoted as D^-= D- $\triangle^-$=D'-$\triangle^+$, δ_D^s represents the support count of s in database D, $|D|$ represents the number of sequences in D. **Incremental sequential pattern mining** is to find all maximal frequent sequences in the database D' given $\triangle^-$, D^-, $\triangle^+$ and the result of mining D.

Suppose that a previous mining has been executed to obtain the supports of frequent sequences. Since the relative order of the sequences within the database does not affect the mining results, we assume that all the deleted sequences are located at the beginning of the database while the new sequences are appended at the end. We also define $b_X^s = min_{s'}\delta_X^{s'}$ for any sequence s and database X, where $(s' \subseteq s) \wedge (|s'| = |s| - 1)$. That means, if s is a k-sequence, b_X^s is the smallest

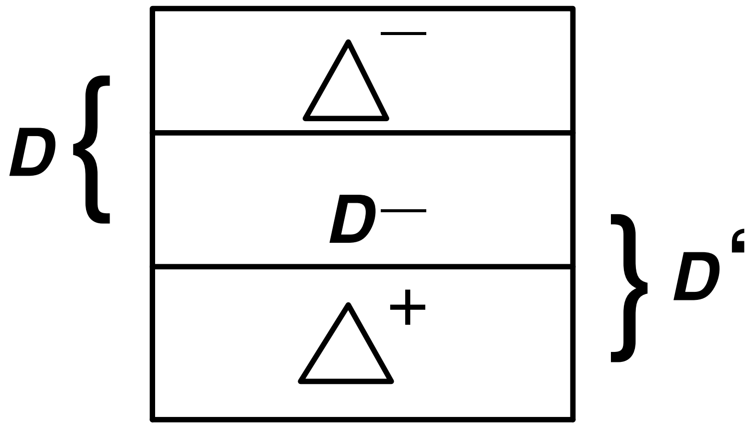

Fig. 3. Definition of D, D', $\triangle^{-}, \triangle^{+}$ and D^{-}.

support count of the (k-1)-subsequences of s in the database X. Since the support count of sequence s can not be larger than the support count of any subsequences of s, b^s_X is an upper bound of δ^s_X. Both GSP+ and MFS+ are based on this property, they aim to reduce the number of candidates, by applying this pruning techniques into the existing algorithms [Zhang et al. 2002].

GSP+ has the same structure with GSP, in each iteration candidate sequences are generated by the same function based on frequent sequences generated in the previous pass. GSP+ employs different pruning techniques and it scans only part of the updated database to check the support of those candidates. The pruning techniques are derived from the following two lemmas.

Lemma 1: If a sequence s is frequent in D', then $\delta^s_D + b^s_{\triangle^+} \geq \triangle^s_D + b^s_{\triangle^+} - b^s_{\triangle^-} \geq |D'| \times \rho_s$. Given a sequence s, if s is a pattern in D, supposed we stored all δ^s_D, by scanning $\triangle^+$ we can get $b^s_{\triangle^+}$, if $\delta^s_D + b^s_{\triangle^+}$ is less than $|D'| \times \rho_s$, we can determine that s is not frequent in D', otherwise we can scan $\triangle^-$ to get $\triangle^s_D + b^s_{\triangle^+} - b^s_{\triangle^-}$, if it is less than $|D'| \times \rho_s$ we can also conclude that s is infrequent in D'. During this pruning process we can avoid scanning D^- and sometimes $\triangle^-$. However if δ^s_D is not available, which means s is not frequent in D, then we turn to Lemma 2.

Lemma 2: If a sequence s is frequent in D' but not in D, then $b^s_{\triangle^+} \geq b^s_{\triangle^+} - \delta^s_{\triangle^-} \geq \delta^s_{\triangle^+} - \delta^s_{\triangle^-} > (|\triangle^+| - |\triangle^-|) \times \rho_s$. Given a candidate sequence s that is not frequent in the old database D, first by scanning $\triangle^+$ we can get $b^s_{\triangle^+}$, if it is less than $(|\triangle^+| - |\triangle^-|) \times \rho_s$ then s can not be frequent in D', otherwise we scan $\triangle^-$ to find $b^s_{\triangle^+} - \delta^s_{\triangle^-}$ if this value is less than $(|\triangle^+| - |\triangle^-|) \times \rho_s$ s also can not be frequent in D'.

By using the pruning techniques based on the aforementioned lemmas, the number of candidate sequences becomes much smaller and the unchanged sequences, which is typically larger than the deleted and inserted sequences, are scanned only

L^{DB}	Frequent sequences in the original database.
L_1^{db}	Frequent 1-sequences embedded in db and validated on U
candExt	Candidate sequences generated from db.
freqExt	Frequent sequences obtained from *candExt* and validated on U.
freqSeed	Frequent sub-sequences of L^{DB} extended with items of L_1^{DB}.
candInc	Candidate sequences generated by appending sequences of *freqExt* to sequences of *freqSeed*
freqInc	Frequent sequences obtained from *candInc* and validated on U.
L^U	Frequent sequences in the updated database.

Table VIII. Notation for ISE

when the pruning techniques fail.

Those pruning techniques are also used in MFS+. First MFS+ uses the frequent sequences in the old database as an estimation of the set of frequent sequences of the updated database. Those sequences with all the possible 1-sequences are taken as candidate sequences, by scanning $\triangle^-$, D^- and $\triangle^+$ all the support counts of those candidates can be obtained. With the minimal support threshold the set of maximal frequent sequences are put into the set MFSS. In the following iterations candidates are generated from MFSS, and the two pruning techniques are used. Firstly, by scanning $\triangle^+$ some candidates can be pruned. Then $\triangle^-$ is scanned to further prune the candidates, for all those left candidates D^- is scanned. The process iterates until no more candidates can be generated or no frequent sequence can be found.

ISM [Parthasarathy et al. 1999] is an incremental sequential pattern mining algorithm based on the SPADE approach of sequential pattern mining. ISM can not only maintain frequent sequences when the database is updated, also it provides a user-interactive interface for users to modify their constraints such as minimal support, items including and excluding.

ISM supposes that all the frequent sequences in the old database with their support counts, and those sequences in the negative border and their support are available in a lattice. By constructing an Incremental Sequence Lattice(ISL) and exploring its properties, ISM prunes the search space for potential new sequences. At the cost of maintaining a summary data structure, the interactive approach performs several orders of magnitude faster than most sequential pattern mining algorithms.

While ISM only considers sequence appending, ISE [MASSEGLIA et al. 2000] considers both the appending of sequences and inserting of new sequences. When we consider sequence appending, all the previous frequent sequences are still frequent, but when new sequences are inserted some of them may become infrequent with the same minimum support. In [MASSEGLIA et al. 2000] incremental sequential mining is defined as following: Let DB be the original database, db be the incremental database where new customer sequences and transactions are inserted. L^{DB} denotes the set of frequent sequences in DB, the **incremental sequential pattern mining problem** is to find frequent sequences in U=DB$\bigcup$db with respect to the same minimum support.

Suppose the length of maximal frequent sequence in the old database is k, ISE decomposes the mining into two subproblems, for those candidate sequences whose length is bigger than k, the GSP algorithm is applied directly. The notation used in this algorithm is summarized in Table VIII. For those candidate sequences whose length is no bigger than k, the algorithm works in the following way:

In the first pass on db, the support count of individual items are counted. With the previous mining result, we can get a set of frequent sequences, which is not frequent in the old database, denoted by L^db_1. Further 2-*candExts* are generated by joining L^db_1 and checking if they exist in db. By scanning U, frequent 2-sequences, 2-*freqExt*, are generated from 2-*candExts*. Also L^{DB} is retrieved, those sequences that proceed those L_1^{db}, according to the time order, are associated with corresponding sequence. The *freqSeeds* are generated by appending L_1^{db} to those sequences. (i+1)-candExt are generated from i-freqExt, while candInc are generated from freqSeed and freqExt, all those candidates are checked by scanning the updated database until no more candInc or candExt having size$\leq (k+1)$ can be generated.

Two pruning techniques are used to optimize the ISE algorithm, they aim to reduce the number of candidate sequences in a early stage by using current information. While ISE only consider extending the suffix of frequent sequences in the original database, IUS [Zheng et al.] extends both the prefix and suffix of old frequent sequences. IUS also use the negative border that ISM uses, but ISM has no memory management approach. IUS defines a minimum threshold for negative border, only those sequence whose support exceed this threshold are included in the negative border, so IUS requires less memory space. In [Ching-Yao Wang 2001] an algorithm is proposed to maintain sequential patterns while some records are deleted from the database. Sometimes the database changes frequently, the differences of sequential patterns are not worth of running the incremental algorithm. In [Zheng et al. 2002], an experimental method, called TPD(Tradeoff between Performance and Difference) is proposed to determine when to update the sequential patterns.

2.2.10 *Periodic Pattern Analysis.* Periodical pattern analysis is the mining of periodic patterns, it aims to find all the recurring patterns in the time-series databases. Periodic pattern mining can be viewed as an extension of sequential pattern mining by taking durations as a set of partitioned sequences, for example we take all the weather data of every year as a sequence. The problem of mining periodic patterns then can be partitioned into three categories: 1). Mining full periodic patterns, every point in time series contributes to the cyclic behavior of the time series. For example, every day in the week contributes to the cycle of the week. 2). Mining partial periodic patterns, only some but not all the points in time series contribute to a specific periodic behavior of the time series. For example, Mary watches movie from 7:00-9:00 pm every Friday. 3). Mining cyclic or periodic association rules, rules that associate a set of events that occur periodically. For example, based on day-to-day transactions, if afternoon tea is well received between 3:00-5:00 pm in a restaurant, dinner will sell well between 7:00-9:00 pm on weekends.

Most of the full periodic pattern mining problems have been studied in signal analysis and statistics, or transformed into sequential pattern mining [Han and

Kamber 2000]. The most useful and challenge part in sequential pattern mining focuss on mining of partial periodic patterns, since it occurs more commonly in the real world. A handful publication of partial periodic pattern mining [Han et al. 1999b] [Yang et al. 2000] [Elfeky 2000] [Bettini et al. 1998] has been proposed.

The problem of mining partial periodic patterns is introduced in [Han et al. 1999b] by Han. Two algorithms for mining single-period and multiple periods patterns have been proposed. The problem of partial periodic pattern mining is defined as following: Assume that a sequence of n time stamped datasets have been collected in a database. For each time instant i, let D_i be a set of features derived from the dataset collected at that instant. The time series features are represented as, S=$D_1, D_2 \cdots D_n$. The mining of frequent **partial periodic sequential patterns** in a time series is to find, possibly with some restriction, all the frequent patterns of the series for one period or a range of specified periods.

For single-period pattern mining, it aims to find all partial periodic patterns for a given period p, support threshold *minimal support* and confidence threshold *minimal confidence* in time-series S. One approach is directly use the Apriori algorithm to the mining process after the sequence is divided into period segments. The problem of finding frequent sequence is similar to finding the frequent itemsets in association rules mining. The Apriori property is used to prune the candidate of large sequences. The total number of scans in this algorithm is no more than the length of period p. Space needed in this method is 2^{F_1} -1 in the worst case, where F_1 is the number of frequent 1-patterns. Another method is called max-subpattern hit set method. In this algorithm F_1 is used to produce a C_{max} the candidate (frequent) max-pattern. A subpattern of C_{max} is **hit set** in a period segment S_i of S if it is the maximal subpattern of C_{max} in S_i. The **hit set**, H, of a time series S is the set of all hit subpatterns in C_{max} in S. During the first scan, as in Apriori, the frequent 1-pattern F_1 is generated. In the second scan, hit set of each period segment is generated and stored into the hit set buffer in a tree structure with its count. The frequent patterns are generated from the hit set by their counts. In this approach only 2 scans of the database are needed, the space needed at most is min{m, 2^{F_1} -1 }, where m is the total number of periods in S.

To find multiple periods based partial periodic patterns, looping over single period using the hit set based approach is one of the naive methods. The algorithm is to directly apply the max-pattern hit set method to each period P_j and the sequence S. Obviously the number of scans in this method is $2 \times k$ since there are k iterations of the sequence, where k is the number of periods in the specified range, while the space needed is $\sum_{j=1}^{k} min\{m_j, 2^{F1j}\}$. Another method for multiple periods is the approach of shared mining. This method is quite similar to the max-pattern single period mining algorithm. During the first scan of the sequence for all periods p_j the frequent 1-pattern $F_{1(p_j)}$and candidate max-pattern $C_{max}(p_j)$ are generated. During the second scan the hit sets of all the periods are generated as in the second scan of max-pattern hit set method. The problem of max-subpattern tree construction and derivation of frequent patterns from the max-subpattern tree is also discussed.

Experiments have been conducted on synthetic time-series database, both for the single period pattern mining and the multiple periods pattern mining the algorithms using max-subpattern hit set outperformed the algorithms based on the Apriori. The reason is obvious with regard to the number of scans over the time series and the space needed. While the max-subpattern based algorithms only scan the database twice the Apriori based algorithms scan the database many time, since the mining object is very large database, the Apriori based algorithms need a large amount of extra disk-I/O operations.

Since time series databases are bound to change over time, incremental mining of partial periodic patterns is introduced in [Elfeky 2000]. With new data added in the database, the periodic patterns may change accordingly. Suppose the frequent patterns and F_1 of the old database is stored physically, the incremental mining algorithm works as follow: 1). Scan the new data to get the frequent 1-patterns, by combining the new 1-patterns with those in the old database, we can form the new max-subpattern C_{max} of the updated database. 2). If the new C_{max} does not change, then the sup-pattern tree will be the same for the new database. 3). If the new C_{max} changed, there are two cases: deleted letters and inserted letters. For deleted letters: the newly added letters of the new C_{max} are deleted and form a C_{max1} that is the subpattern of the old one, the subpart of the old subpattern trees that contain C_{max1} are inserted to the new subpattern tree. For inserted letters: Using the new C_{max} as root of the tree and scan the old database to increase the count of hit patterns that contain newly inserted letters and decrease the count of hit patterns without inserted letters. 4). Scan the new data and update the max-subpattern tree, the step is similar to the second step in the basic algorithm mentioned above. 5). Traverse the tree to determine the actual count of each pattern and determine the frequent patterns. The algorithm of mining the combination of two mined database is also introduced based on the above algorithm.

Most researches mentioned above focused on mining synchronous periodic patterns, but in practice some periodic patterns can not be recognized because of presence of random noisy and disturbance. Mining asynchronous periodic pattern in time series data is introduced in [Yang et al. 2000] to find those patterns that occur frequently in some subsequences but their occurrences may be shifted with disturbance. Two parameters min_rep and max_dis are used to specify the minimum number of occurrences that is required within each subsequences and the maximum disturbance between any two successive subsequences. Related research about mining sequential pattern in a noisy environment has been explored in [Yang et al. 2002], the concept of compatibility matrix is introduced to provide a probabilistic connection from observed values to the true values, based on the matrix the real support of a pattern can be captured.

3. CONCLUSION

In this paper, we surveyed the list of existing association rule mining techniques. This investigation is prepared to our new project titled *mining historical changes to web delta.*

REFERENCES

Agrawal, R., Imielinski, T., and Swami, A. N. 1993. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, P. Buneman and S. Jajodia, Eds. Washington, D.C., 207–216.

Agrawal, R. and Srikant, R. 1994. Fast algorithms for mining association rules. In *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, J. B. Bocca, M. Jarke, and C. Zaniolo, Eds. Morgan Kaufmann, 487–499.

Agrawal, R. and Srikant, R. 1995. Mining sequential patterns. In *Eleventh International Conference on Data Engineering*, P. S. Yu and A. S. P. Chen, Eds. IEEE Computer Society Press, Taipei, Taiwan, 3–14.

Berkhin, P. 2002. Survey of clustering data mining techniques. Tech. rep., Accrue Software, San Jose, CA.

Bettini, C., Wang, X. S., and Jajodia, S. 1998. Mining temporal relationships with multiple granularities in time sequences. *Data Engineering Bulletin 21,* 1, 32–38.

Beyer, K. and Ramakrishnan, R. 1999. Bottom-up computation of sparse and Iceberg CUBE. 359–370.

Chen, M.-S., Han, J., and Yu, P. S. 1996. Data mining: an overview from a database perspective. *Ieee Trans. On Knowledge And Data Engineering 8*, 866–883.

Ching-Yao Wang, Tzung-Pei Hong, S.-S. T. 2001. Maintenance of sequential patterns for record deletion. *ICDM*, 536–541.

Duda, R. and Hart. 1973. *Pattern Classification and Scene Analysis.* Wiley&Sons, Inc.

Elfeky, M. G. 2000. Incremental mining of partial periodic patterns in time-series databases. Tr, Information Assurance and Security, Purdue University.

Garofalakis, M. N., Rastogi, R., and Shim, K. 1999. Spirit: Sequential pattern mining with regular expression constraints. In *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK*, M. P. Atkinson, M. E. Orlowska, P. Valduriez, S. B. Zdonik, and M. L. Brodie, Eds. Morgan Kaufmann, 223–234.

Han, J., Dong, G., and Yin, Y. 1999a. Efficient mining of partial periodic patterns in time series database. In *Fifteenth International Conference on Data Engineering.* IEEE Computer Society, Sydney, Australia, 106–115.

Han, J., Dong, G., and Yin, Y. 1999b. Efficient mining of partial periodic patterns in time series database. In *Fifteenth International Conference on Data Engineering.* IEEE Computer Society, Sydney, Australia, 106–115.

Han, J. and Kamber, M. 2000. *Data Mining Concepts and Techniques.* Morgan Kanufmann.

Han, J., Koperski, K., and Stefanovic, N. 1997. GeoMiner: a system prototype for spatial data mining. 553–556.

Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., and Hsu, M.-C. 2000. Freespan: frequent pattern-projected sequential pattern mining. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM Press, 355–359.

James, M. 1985. *Classification Algorithms.* Wiley&Sons, Inc.

Joshi, M., Karypis, G., and Kumar, V. 2001. A universal formulation of sequential patterns. In *Proceedings of the KDD'2001 workshop on Temporal Data Mining.*

Koperski, K. and Han, J. 1995. Discovery of spatial association rules in geographic information databases. In *Proc. 4th Int. Symp. Advances in Spatial Databases, SSD*, M. J. Egenhofer and J. R. Herring, Eds. Vol. 951. Springer-Verlag, 47–66.

Koperski, K. and Han, J. 1996. Data mining methods for the analysis of large geographic databases.

Lin, M.-Y. and Lee, S.-Y. 2002. Fast discovery of sequential patterns by memory indexing. In *Proc. of 2002 DaWaK.* 150–160.

Mannila, H., Toivonen, H., and Verkamo, A. I. 1995. Discovering frequent episodes in sequences. In *Interantional Conference on Knowledge Discovery and Data Mining.* IEEE Computer Society Press.

MASSEGLIA, F., PONCELET, P., and TEISSEIRE, M. 2000. Incremental mining of sequential patterns in large databases.

Murthy, S. K. 1998. Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Mining and Knowledge Discovery 2,* 4, 345–389.

Park, J. S., Chen, M.-S., and Yu, P. S. 1995. An effective hash based algorithm for mining association rules. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, M. J. Carey and D. A. Schneider, Eds. San Jose, California, 175–186.

Parthasarathy, S., Zaki, M. J., Ogihara, M., and Dwarkadas, S. 1999. Incremental and interactive sequence mining. In *CIKM.* 251–258.

PEI, J., HAN, J., PINTO, H., CHEN, Q., DAYAL, U., AND HSU, M. C. 2001. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. Int. Conf. on Data Engineering.

PINTO, H., HAN, J., PEI, J., WANG, K., CHEN, Q., AND DAYAL, U. 2001. Multi-dimensional sequential pattern mining. In *CIKM*. 81–88.

SRIKANT, R. AND AGRAWAL, R. 1996. Mining sequential patterns: Generalizations and performance improvements. In *Proc. 5th Int. Conf. Extending Database Technology, EDBT*, P. M. G. Apers, M. Bouzeghoub, and G. Gardarin, Eds. Vol. 1057. Springer-Verlag, 3–17.

YANG, J., WANG, W., AND YU, P. S. 2000. Mining asynchronous periodic patterns in time series data. In *Knowledge Discovery and Data Mining*. 275–279.

YANG, J., WANG, W., AND YU, P. S. 2001. Infominer: mining surprising periodic patterns. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM Press, 395–400.

YANG, J., WANG, W., YU, P. S., AND HAN, J. 2002. Mining long sequential patterns in a noisy environment. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*. ACM Press, 406–417.

ZAKI, M. J. 2001. SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning 42,* 1/2, 31–60.

ZHANG, M., KAO, B., CHEUNG, D. W., AND YIP, C. L. 2002. Efficient algorithms for incremental update of frequent sequences. In *PAKDD*. 186–197.

ZHANG, M., KAO, B., YIP, C., AND CHEUNG, D. 2001. A gsp-based efficient algorithm for mining frequent sequences. In *Proc. 2001International Conference on Artificial Intelligence (IC-AI 2001)*.

ZHENG, Q., XU, K., AND MA, S. 2002. When to update the sequential patterns of stream data.

ZHENG, Q., XU, K., MA, S., AND LV, W. The algorithms of updating sequential patterns.