Experimental Results on a Constrained Based Sequential Pattern Mining for telecommunication alarm data

Jain-Zhi Ouh, Pei-Hsin Wu, and Ming-Syan Chen

Department of Electrical Engineering

National Taiwan University

Taipei, Taiwan, ROC

E-mail: {mschen@cc.ee.ntu.edu.tw, alex, peggywu@arbor.ee.ntu.edu.tw}

Abstract

A telecommunication system produces daily a large amount of alarm data which contains hidden valuable information about the system behavior. The knowledge discovered from alarm data can be used in finding problems in networks and possibly in predicting severe faults. In this paper, we devise a solution procedure for mining sequential alarm patterns from the alarm data of a GSM system. First, by observing the features of the alarm data, we develop operations for data cleaning without compromising the quality of sequential alarm patterns obtained. After the data cleaning procedure, we transform the alarm data into a set of alarm sequences. Note that the consecutive alarm events exist in the alarm sequences, and it is complicated to count the occurrence counts of events and extract patterns. Hence, we devise a new counting method to determine the occurrence count of the sequential alarm patterns in accordance with the nature of alarms. More importantly, by utilizing time constraints to restrict the time difference between two alarm events, we devise a mining algorithm to discover useful sequential alarm patterns. The proposed mining algorithm is implemented and applied to test against a set of real alarm data provided by a cellular phone company. The quality of knowledge discovered is evaluated. The experimental results show that the proposed operations of data cleaning are able to improve the execution of our mining algorithm significantly and the knowledge obtained from the alarm data is very important from the perspective of network operators for alarm prediction and alarm control.

Keywords —Data mining, sequential alarm patterns, alarm data, telecommunication databases.

Figure 1. The general architecture of a GSM system.

1 Introduction

Due to recent technology advances, an increasing number of users are accessing various information system via wireless communication. Such information systems as stock trading, wireless conferencing, are being provided by information services and application providers [8][9][13][14], and mobile users are able to access such information via wireless communication from anywhere at anytime [4][7][19].

The architecture of wireless networks varies from one standard to another. Use of the *Global System for Mobile communication* (GSM) continues to spread throughout the world. Figure 1 shows the general architecture of the GSM system. It can be seen that a GSM system comprises lots of interconnected components and each component also contains several subcomponents. Individual components and subcomponents generate alarms indicating some sort of abnormal situations. Table 1 shows an example of selected alarm data where sourceID is the identification of an individual component that generates the alarm and errorID in-

icates the abnormal situation. Thus, a GSM system produces daily a large amount of alarm data which contains hidden valuable information about the system behavior. The knowledge discovered from alarm data can be used in finding problems in networks and possibly in predicting severe faults. Since the number of alarms could be in tens of thousands daily, it is infeasible to investigate these alarms manually. In order to extract the valuable knowledge from the alarm data, we utilize the techniques of data mining in this study.

day	time	source	sourceID	errorID	...
2001/02/01	15:07:09	SITE	S1	A	...
2001/02/01	15:46:23	SITE	S5	C	...
2001/02/01	19:04:37	BSC	S3	D	...
⋮	⋮	⋮	⋮	⋮	...

Table 1 An example of selected alarm data.

Data mining, which is also referred to as *knowledge discovery in databases*, means a process of extracting implicit, previously unknown and potentially useful information (in terms of knowledge rules, constraints, regularities) from data in databases. By knowledge discovery in databases, interesting knowledge, regularities, or high-level information can be extracted from the relevant sets of data in databases and be investigated from different angles. As pointed out in [5], mining information and knowledge from large databases has been recognized by many researchers as a key research topic in database systems and machine learning, and by many industrial companies as an important area with an opportunity of major revenues. The discovered knowledge can be applied to information management, query processing, decision making, process control, and many other applications. Furthermore, several emerging applications in information providing services, such as on-line services, World Wide Web, and telecommunication systems, also call for various data mining techniques to better understand user behavior, to meliorate the service provided, and to increase the business opportunities. Notice that various data mining capabilities have been explored in the literature [5]. One of the most important data mining problems is mining association rules in transaction databases [1][2][6][15][18]. Also, mining classification rules is an approach to developing rules that can efficiently classify data items based on certain features [17][20]. Mining sequential patterns is the study to explore the knowledge of ordered data [3][10][16].

It is noted that utilizing the technique of mining sequential patterns is able to extract valuable knowledge from the

Figure 2. An example of sequential alarm patterns

alarm data generated by a GSM system. In mining sequential patterns, the input data is a set of sequences, called *data-sequences*. Each data-sequence is a list of transactions, where each transaction is a set of literals, called *items*. Typically there is a transaction-time associated with each transaction. A *sequential pattern* also consists of a list of sets of items. The problem is to find all sequential patterns with a user-specified minimum *support*, where the support of a sequential patterns is the percentage of data-sequences that contain the pattern. In prior works, the constraint imposed by the time difference between two consecutive events has not been explored. For example [3], the customers typically rent "Star Wars" first, and then "Empire Strikes Back". Note that such a pattern did not contain the time difference of these two renting activities. Hence, the knowledge of time difference has not been taken into account when the mining results are produced. However, such a time difference between two events is in fact an important knowledge when mining the alarm data since, with this knowledge, one can not only judge the relevance of these two events, but also predict the alarm sequence and take proper steps to prevent the occurrence of the alarms if all possible.

An example sequential alarm sequence that we shall explore in the alarm data is shown in Figure 2. The symbol in each circle represents the errorID, same as in Table 1, and $T_{i,j}$ denotes the time difference between alarm event i and alarm event j . As mentioned above, this knowledge is important from the perspective of network operators in that a network operator can hence be warned beforehand of severe alarms and take proper provision. For example, if the network operator detects that the alarm A occurring at time t , he/she should dissipate this alarm before the time $t + T_{A,B}$ to alleviate the abnormal situations incurred. It can be seen that the problem of mining sequential patterns for the telecommunications is intrinsically different from that for the rental industry, and particularly ought to capture the presence of time difference between events. Therefore, it is necessary to develop a new mining algorithm for mining constrained sequential alarm patterns for a telecommunication system.

In this paper, we devise a solution procedure for min-

ing sequential alarm patterns from the alarm data of a GSM system. First, by observing the features of the alarm data, we develop operations for data cleaning without compromising the quality of sequential alarm patterns obtained. Specifically, one operation is to delete those undetermined alarm events and another is to merge the repeated alarm events caused by the same abnormal situation. After the data cleaning procedure, we transform the alarm data into a set of alarm sequences. Note that the consecutive alarm events exist in the alarm sequences, and it is complicated to count the occurrence counts of events and extract meaningful alarm patterns. Hence, we devise a new counting method to determine the occurrence count of the sequential alarm patterns in accordance with the nature of alarms. More importantly, by utilizing time constraints to restrict the time difference between two alarm events, we devise an algorithm of mining sequential alarm patterns (to be referred to as algorithm MSAP) to discover useful sequential alarm patterns. The algorithm MSAP is implemented and applied to test against a set of real alarm data obtained. The quality of knowledge discovered is evaluated. The experimental results show that the proposed operations of data cleaning are able to improve the execution of our mining algorithm significantly and the knowledge obtained from the alarm data is very important from the perspective of network operators for alarm prediction and alarm control.

This paper is organized as follows. Preliminaries are given in Section 2. In Section 3, we devise the constraint-based algorithm for mining sequential alarm patterns. Experimental results are presented in Section 4. This paper concludes with Section 5.

2 Preliminaries

To facilitate the presentation of this paper, some preliminaries are given in this section. In Section 2.1, we describe the alarm data in a GSM system. In Section 2.2, the procedure of mining sequential alarm patterns is presented.

2.1 Alarm Data in GSM

Figure 3 shows the components of a GSM system. The Mobile Services Switching Center (MSC) is a complete exchange system which is able to route calls from the fixed network to an individual mobile station. The MSC has connections to other entities of the GSM system. Those connections allow the MSC to gather the current statuses of mobile stations. The Base Station Controller (BSC) provides the management of the Base Transceiver Station (BTS) and also the communication between the MSC and the BTS.

Figure 3. The components of GSM : BSS, BSC, and BTS.

Figure 4. The procedure of mining sequential alarm patterns.

The BTS supports the radio transmission/reception and the control of radio functions.

When detecting an anomalous situation, the GSM system generates an alarm event to alert the operators for this anomalous situation. All these alarm events are recorded in the alarm log table, which has several attributes, such as day, time, source, sourceID, severity and errorID. The attributes *Day* and *Time* record the time when the alarm event occurred. The attribute *Source* is the type of the component generating the alarm event. As to the other attributes of alarm log table, *SourceID* is the identification of an individual component and *severity* means the severity of the anomalous situation. Each anomalous situation or alarm event is associated with the identification number, denoted as errorID.

2.2 Procedure of Mining Alarm Sequential Patterns

In this paper, we explore the problem of mining sequential alarm patterns from the alarm log data. Figure 4 shows the procedure of mining sequential alarm patterns that involves the following tasks:

1. Data cleaning: In order to extract useful sequential alarm patterns, data cleaning and data preprocessing are needed [11]. The operations of data cleaning include select-

ing those attributes needed for data mining and removing those redundant information or outliers.

2. Discovery of patterns: After the process of data cleaning, we employ a data mining algorithm to discover the valuable sequential alarm patterns from the alarm data. The sequential alarm patterns contain not only the temporal relationship but also the time spans among the alarm events.

With the problem and the corresponding tasks described above, we then develop an approach to mining sequential alarm patterns, which is described in Section 3.

3 Mining Sequential Alarm Patterns

In this section, we describe the process of data cleaning in Section 3.1. In Section 3.2, we devise an algorithm of mining sequential alarm patterns (to be referred to as algorithm MSAP) to extract useful sequential alarm patterns.

3.1 Data Cleaning

The goal of data cleaning is to identify the available data sources and extract the data that is needed for preliminary analysis in preparation for future mining. Clearly, the process of data cleaning depends on the patterns to be mined. Note that the alarm log data contains not only useful information but also dummy information. Before mining sequential alarm patterns, the procedure of data cleaning is needed. Specifically, we have the following two operations for data cleaning:

Cleaning undetermined alarm events

Some anomalous causes that cannot be recognized by the alarm system are assigned their errorID to be *UID* and the alarm event with its errorID *UID* is frequently found in the alarm log data. Because of the frequent occurrence of the alarm event *UID*, the sequential alarm patterns are those patterns containing lots of the alarm event *UID*. Clearly, these sequential alarm patterns are of less interest. Thus, we shall clean undetermined alarm events to improve the quality of the sequential alarm patterns mined.

Merging repeated alarm events

It is also noted that some repeated alarms appear frequently in the alarm log data. It can be verified that some repeated alarms are independent events and the others are caused by the same abnormal situations. For example, the alarm log data has the sequence $\langle AAAABC \rangle$ in which alarm event A repeats four times. Notice that if these repeated alarm events are from the same abnormal situation, we would like to merge them to one alarm event (i.e., the $\langle AAAABC \rangle$ becomes $\langle ABC \rangle$). Otherwise, we retain the repeated alarms. Note that not all repeated alarm events could

be merged. Merging the repeated alarm events should be done in accordance with the *ceased table* in the telecommunication database system. The *ceased table* contains the time when an alarm event is generated and the time when the alarm event is ceased.

3.2 Algorithm MSAP: Mining Sequential Alarm Patterns

After the data cleaning procedure, we transform the alarm data into a set of alarm sequences. Then, we devise an algorithm MSAP (standing for Mining Sequential Alarm Patterns) to extract useful sequential alarm sequences. Denote the set of errorID as E and define an alarm event as a pair (eid, t) , where $eid \in E$ is an errorID and t is the time when the alarm event occurred. For example, an alarm event (A, 2001/2/10 16:14:08) means that the alarm A occurred at 16:14:08 on 10 Feb in 2001.

Definition 1: An alarm sequence S_{sid} is a sequence of ordered alarm events generated by the component *sid*, where *sid* is the sourceID and $S_{sid} = \langle (eid_1, t_1), (eid_2, t_2), (eid_3, t_3), \dots, (eid_n, t_n) \rangle$, where $t_i < t_j$ if $i < j$.

As described before, the alarm events are generated from individual components of the GSM. In this paper, we address the problem of mining sequential alarm patterns for the individual components and create an alarm sequence table containing alarm sequences. Table 2 is an alarm sequences table where the component of GSM is site (i.e., the base station) and the duration time of the alarm sequences is two hours, from 0:00am to 2:00am. Four alarm sequences (i.e., S_{2001} , S_{2002} , S_{2003} and S_{2004}) are given in Table 2. The *length* of an alarm sequence S_{sid} , denoted as $|S_{sid}|$, is the number of events in the sequence *sid*. A sequence of length k is called a k -sequence. It can be seen that the length of S_{2001} is four.

As pointed out earlier, the knowledge of time difference has not been taken into account in prior works for sequential pattern mining. However, such a time difference between two events is in fact an important knowledge when mining the alarm data since, with this knowledge, one can predict the alarm sequence and take proper steps to prevent the occurrence of the alarms if all possible. In order to find the critical sequential alarm patterns concerned by network operators, algorithm MSAP allows network operators to set the restriction on the time difference between two alarm events in sequential alarm patterns. Thus, we have the following definition.

Definition 2: *Urgent window*, denoted as δ , is a user-specified time interval. We only count the candidate C_2 ,

if the alarm events occur within the interval of an urgent window δ .

site ID	Alarm sequences
2001	$\langle (A, 0:12) (C, 0:25) (D, 1:07) (C, 1:19) \rangle$
2002	$\langle (C, 0:37) (C, 0:42) (B, 1:11) (A, 1:36) \rangle$
2003	$\langle (B, 0:05) (C, 0:21) (C, 0:54) (A, 1:25) \rangle$
2004	$\langle (B, 0:27) (C, 0:37) (A, 1:22) \rangle$

Table 2. An example of alarm sequences.

It will be seen that the size of an urgent window is set to determine the maximal time lag two events are allowed if they are to be viewed relevant to each other. Next, algorithm MSAP is devised and applied to alarm sequences to discover sequential alarm patterns. C_k is a set of candidate k -sequences. The occurrence count of C_k in alarm sequences is referred to as support. Let L_k represent the set of large k -sequences, where there are a sufficient number of alarm sequences containing this k -sequence. Such a threshold number is called minimum support in this paper. Using a candidate generation procedure, algorithm MSAP makes multiple scans over the set of alarm sequences to generate the possible sequential alarm patterns.

By observing the alarm sequences, the processing of counting support is nontrivial. For the customer transaction data, the support is increased only once per customer even if the customer buys the same set of items in two different transactions. In contrast, alarm event A and alarm event C could occur several times in one day. As a consequence, it is essential to develop the counting method to obtain the support of these candidate alarm sequences.

In order to find useful sequential alarm patterns, we only consider the event pair (i, j) , where i and j are alarm events and alarm j is the closest to alarm event i .

Figure 5 shows the example of counting the support of $\langle AC \rangle$. Note that we only include for counting $\langle AC \rangle$ whose interval is the shortest, i.e., there is no other $\langle AC \rangle$ in the interval. Therefore, only $\langle (A, 0:12)(C, 0:25) \rangle$ and $\langle (A, 1:25)(C, 1:42) \rangle$ are counted by algorithm MSAP. Thus, in this example, the support of $\langle AC \rangle$ is two.

With the above counting process, the large k -sequence has the following property:

Property 1: For any sequential alarm pattern $sid_i = \langle r_1, r_2, \dots, r_k \rangle$ in L_k , the support of this sequential alarm sequence is smaller than that of the sequential alarm patterns $sid_j = \langle q_1, q_2, \dots, q_{k-1} \rangle$ in L_{k-1} , where $q_x = r_x$ and $1 \leq x \leq k-1$. For example, the support of $\langle ABDF \rangle$ is smaller than that of $\langle ABD \rangle$.

In light of Property 1, we devise *Procedure Candidate_Generation* to generate the possible candidate k -

Figure 5. The occurrence count of the sequential alarm sequence $\langle AC \rangle$

sequences and algorithm MSAP to mine sequential alarm patterns. This solution procedure is outlined as follows.

Procedure Candidate_Generation

```

insert into  $C_k$ 
select  $p.event_1, p.event_2, \dots, p.event_{k-1}, q.event_{k-1}$ 
from  $p \in L_{k-1}, q \in L_{k-1}$ 
where  $p.event_2 = q.event_1, \dots, p.event_{k-1} = q.event_{k-2}$ 
for all  $c \in C_k$  do
    for all  $(k-1)$ -subpattern  $s$  of  $c$  do
        if ( $s \notin L_{k-1}$ ) then
            delete  $c$  from  $C_k$ 

```

Algorithm MSAP

Input: A set of alarm sequences $\{S\}$, urgent window δ , minimum support sup

Output: sequential alarm patterns

1. $L_1 = \{\text{large 1-sequence}\}$
2. $C_2 = \text{new candidates generated from } L_1$
by calling *procedure Candidate_Generation*
3. **for each** alarm-sequence s in the database **do**
4. **for each** candidates $c \in C_2$ **do**
5. search c in sequence s
6. **if** find an event pair p matched $c \in C_2$
and the time-difference of event pair p
are less than the urgent window δ
7. increase the count of $c \in C_2$.
and record the time-difference
8. $L_2 = \text{candidates in } C_2 \text{ with minimum support required}$
9. Compute the mean μ and standard deviation σ of
the time-difference of each $l \in L_2$
10. **for** ($k = 3; L_{k-1} \neq \emptyset; k++$) **do**
11. **begin**
 $C_k = \text{new candidates generated from } L_{k-1} \text{ by}$
 calling *procedure Candidate_Generation*
12. **for each** alarm-sequence s in the database **do**
13. **for each** candidates $c \in C_k$ **do**
14. search c in sequence s
15. **if** find a pattern x matched c and
the time difference of each event
pair is less than or equal to δ
16. increase the count of
the candidate $c \in C_k$
17. $L_k = \text{candidates in } C_k \text{ with minimum support required}$
18. **end**

Note that L_1 can be simply generated from the counting of appearing alarm events. Then, C_2 are obtained by the procedure of Candidate_Generation (line 2 of algorithm MSAP). After generating possible C_2 , the occurrence count and the time difference of all candidate $c \in C_2$ are obtained by scanning all the alarm sequences (from line 3 to line 7 in algorithm MSAP). Algorithm MSAP calculates the mean and standard deviation of time-difference for each $l \in L_2$ (line 9 of algorithm MSAP). L_2 can be determined by proper trimming on C_2 . By utilizing procedure Candidate_Generation, C_k can be generated by $L_{k-1} * L_{k-1}$ where $*$ is an operation for concatenation (from line 10 to line 18 in algorithm MSAP). The support of each k -sequences is determined for the identification of L_k .

3.3 An Illustrative Example for Mining Sequential Alarm Patterns

Consider an example alarm sequence given in Table 2. In each iteration (or each pass), algorithm MSAP constructs a candidate set of large sequences, counts the number of occurrences of each candidate sequence, and then determines large sequences based on a pre-determined minimum support. In the first iteration, algorithm MSAP simply scans all the alarm sequences to count the number of occurrences for each alarm event. The set of candidate 1-sequence, C_1 , obtained is shown in Table 3a. Assume that the minimum support required is 2. The set of large 1-sequences, L_1 , composed of candidate 1-sequences with the minimum support required, can then be determined.

C_1	support		L_1	support
A	4	$\Rightarrow$	A	4
B	3		B	3
C	7		C	7
D	1			

(a) Generation of C_1 and L_1 .

C_2	support	Time difference	μ	σ
AA	0		0	0
AB	0		0	0
AC	1	13	13	0
BA	2	25, 55	40	15
BB	0		0	0
BC	2	16, 10	13	3
CA	3	54, 31, 45	43.33	11.59
CB	1	29	29	0
CC	3	54, 5, 33	30.67	20.07

$\Rightarrow$

L_2	support
BA	2
BC	2
CA	3
CC	2

(b). Generation of C_2 and L_2 .

C_3	support		L_3	support
BCA	2	$\Rightarrow$	BCA	2
BCC	1			
CCA	2		CCA	2
CCC	0			

(c). Generation of C_3 and L_3 .

Table 3: Generation of candidate sequences and large sequences.

To discover the set of large 2-sequence, algorithm MSAP uses $L_1 * L_1$ to generate a candidate set of sequences C_2 . Next, the four alarm sequences in Table 2 are scanned. The support of each candidate sequence in C_2 whose time difference between two alarm events is smaller than δ (i.e., the urgent window) is counted. Table 3b represents the result from such counting in C_2 . As described before, time plays an important role in mining sequential alarm sequences. Thus, algorithm MSAP calculates the mean and the variance of time difference for each 2-sequence in C_2 . The set of large 2-sequence, L_2 , is therefore determined based on the support of each candidate 2-sequence in C_2 . The set of candidate 3-sequence is generated from $L_2 * L_2$. Algorithm MSAP then scans the alarm sequences and discovers the large 3-sequences in Table 3c. Following the same procedure, algorithm MSAP discovers the large k -sequences. If there is no other large k -sequence discovered, algorithm MSAP stops and those large k -sequences are the sequential alarm patterns mined from the alarm data.

4 Experimental Results

The effectiveness of mining sequential alarm sequences on real alarm data is evaluated in this section. Experimental results of utilizing data cleaning techniques are shown in Section 4.1. Then, we present the experimental results of algorithm MSAP in Section 4.2.

4.1 The Impact of Data Cleaning

As mentioned before, the alarm log data contains not only useful information but also dummy information, thus

Figure 6. The performance improvement while performing the operations of data cleaning.

calling for data cleaning. The execution time of mining sequential alarm patterns can also be reduced. We now examine the impact of data cleaning. We select one week alarm log data from the alarm database, where the number of cells is 6844, the number of sites is 3830 and the number of BSC is 253.

In the first place, the performance improvement of MSAP is computed as followings:

$$= \frac{\text{performance improvement}}{\text{execution time before cleaning} - \text{execution time after cleaning}} \times \text{execution time before cleaning}$$

We then investigate the performance improvement when the operations of data cleaning are applied to the alarm data. Figure 6 shows the performance improvement while performing the procedure of data cleaning in advance. It can be seen that the execution times of algorithm MSAP vary due to the reason that the numbers of alarm events generated are different. For the process of merging repeated alarm events, the amounts of repeated alarms are small. It can be seen in Figure 6 that the performance improvement is dominated by the operation of cleaning undetermined alarm events. Notice that the execution of algorithm MSAP after data cleaning is much faster than that without data cleaning, showing the merit of the data cleaning procedure.

4.2 Experimental Results of Algorithm MSAP

In this section, we evaluate the correctness of the sequential alarm patterns discovered by algorithm MSAP. Without

loss of generality, we select three months alarm log data, and set the value of the minimum support to be 100 and that of the urgent window to be ten minutes. Table 4 shows the selected sequential alarm patterns mined by algorithm MSAP and the description of errorID is shown in Table 5.

L_2
$\langle(A)(B)\rangle$
$\langle(B)(C)\rangle$
$\langle(B)(D)\rangle$

(a). The selected large 2-sequences (L_2).

L_3
$\langle(A)(B)(C)\rangle$
$\langle(B)(C)(D)\rangle$

(b). The large 3-sequences (L_3).

Table 4. The selected sequential alarm patterns discovered by algorithm MSAP.

errorID	Description
A	LPDL link down
B	LPDL link transition
C	Synchronization changed to holdover mode
D	End of holdover mode

Table 5. Description of errorID.

It is worth mentioning that the sequential alarm patterns are beneficial for managing and designing alarm systems. With the sequential alarm patterns discovered, one can predict the alarm sequence and take the proper steps to prevent the occurrence of alarms if all possible. Furthermore, as pointed out in [12], alarm filtering requires the knowledge of the alarm sequences. By combining and transforming the related alarm events into one alarm, one is able to design proper alarm systems for telecommunication systems. Clearly, these sequential alarm patterns mined by our algorithm MSAP are also very important in these design issues.

5 Conclusions

In this paper, we devised a solution procedure for mining sequential alarm patterns from the alarm data of a GSM system. First, by observing the features of the alarm data, we developed operations for data cleaning without compromising the quality of sequential alarm patterns obtained. After the data cleaning procedure, we transformed the alarm data

into a set of alarm sequences. We devised a new counting method to determine the occurrence count of the sequential alarm patterns in accordance with the nature of alarms. By utilizing time constraints to restrict the time difference between two alarm events, we devised algorithm MSAP to obtain the mean and the standard deviation of two alarm events, and discover useful sequential alarm patterns. The quality of knowledge discovered is evaluated. The experimental results showed that the proposed operations of data cleaning are able to improve the execution of our mining algorithm and the knowledge obtained from the alarm data is very important from the perspective of network operators for alarm prediction and alarm control.

References

- [1] R. Agrawal, T. Imielinski, and A. Swami. Mining Associations between Sets of Items in Massive Databases. In *Proceeding of ACM SIGMOD*, pages 207–216, May 1993.
- [2] R. Agrawal and J. Shafer. Parallel Mining of Association Rules. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):866–883, December 1996.
- [3] R. Agrawal and R. Srikant. Mining Sequential Patterns. *Proceedings of the Eleventh IEEE International Conference on Data Engineering*, pages 3–14, 1995.
- [4] B. Bruegge and B. Bennington. Applications of Mobile Computing and Communication. *IEEE Personal Communication*, pages 64–71, February 1996.
- [5] M. S. Chen, J. Han, and P. S. Yu. Data Mining: An Overview from Database Perspective. *IEEE Transactions on Knowledge and Data Engineering*, (6):866–883, December 1996.
- [6] D. W. Cheung, V. T. Ng, W. Fu, and Y. Fu. Efficient Mining Association Rules in Distributed Databases. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):911–922, December 1996.
- [7] N. Davies, G. S. Blair, K. Cheverst, and A. Friday. Supporting Collaborative Application in a Heterogeneous Mobile Environment. *Computer Communication Special Issues on Mobile Computing*, 1996.
- [8] M. H. Dunham. Mobile Computing and Databases. *Tutorial of International Conference on Data Engineering*, February 1998.
- [9] A. Elmagarmid, J. Jain, and T. Furukawa. Wireless Client/Server Computing for Personal Information Services and Applications. *ACM SIGMOD RECORD*, 24(4):16–21, December 1995.
- [10] J. Han, G. Dong, and Y. Yin. Efficient Mining of Partial Periodic Patterns in Time Series Database. In *Proceeding of the 15th International Conference on Data Engineering*, March 1999.
- [11] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Academic Press, 2001.
- [12] K. Hätönen, M. Klemettinen, H. Mannila, P. Ronkainen, and H. Toivonen. Knowledge discovery from telecommunication network alarm databases. In *Proceedings of the Twelfth International Conference on Data Engineering, February 26 - March 1, 1996, New Orleans, Louisiana*, pages 115–122, 1996.
- [13] N. Krishnakumar and R. Jain. Escrow Techniques for Mobile Sales and Inventory Applications. *ACM Journal of Wireless Network*, 3(3):235–246, July 1997.
- [14] D. L. Lee. Data Management in a Wireless Environment. *Tutorial of International Conference on Database System for Advance Applications*, April 1999.
- [15] J.-L. Lin and M. H. Dunham. Mining Association Rules: Anti-Skew Algorithms. In *Proceeding of the 14th International Conference on Data Engineering*, pages 486–493, February 1998.
- [16] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of Frequent Episodes in Event Sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1996.
- [17] R. Ng and J. Han. Efficient and Effective Clustering Methods for Spatial Data Mining. *Proceedings of the 18th International Conference on Very Large Data Bases*, pages 144–155, September 1994.
- [18] J.-S. Park, M.-S. Chen, and P. S. Yu. Using a Hash-Based Method with Transaction Trimming for Mining Association Rules. *IEEE Transactions on Knowledge and Data Engineering*, 9(5):813–825, October 1997.
- [19] M. Satyanarayanan. Mobile Information Access. *IEEE Personal Communication*, pages 26–33, February 1996.
- [20] J. Shafer, R. Agrawal, and M. Mehta. SPRINT: A Scalable Parallel Classifier for Data Mining. In *Proceeding of the 22th International Conference on Very Large Databases*, September 1996.