

Incremental Mining for Temporal Association Rules for Crime Pattern Discoveries

Vincent Ng*, Stephen Chan, Derek Lau, Cheung Man Ying

Department of Computing
The Hong Kong Polytechnic University,
Hong Kong, China

cstyng@comp.polyu.edu.hk

Abstract

In recent years, the concept of temporal association rule (TAR) has been introduced in order to solve the problem on handling time series by including time expressions into association rules. In real life situations, temporal databases are often appended or updated. Rescanning the complete database every time is impractical while existing incremental mining techniques cannot deal with temporal association rules. In this paper, we propose an incremental algorithm for maintaining temporal association rules with numerical attributes by using the negative border method. The new algorithm has been implemented to support the discoveries of crime patterns in a district of Hong Kong. We have also experimented with another real life database of courier records of a shipping company. The preliminary results show a significant improvement over rerunning TAR algorithm..

Keywords: Temporal Association Rules, Incremental Mining, Crime Analysis.

1 Introduction

Association rule plays an important role in data mining and has been becoming applicable in many areas. Since the pioneering works of Agrawal, Imielinski and Swami (1993) and Agrawal and Srikant (1994), it has led to many proposals of mining of association rules such as fast mining approaches, updating approaches, and various formations of rule patterns such as temporal patterns discussed in Wang, Yang and Muntz (1999, 2001). In general, most transactional database systems accumulate small size of incremental datasets and are appended into the main databases regularly. In such situations, incremental mining becomes a necessity. Incremental mining algorithms (Cheung, Han, Ng, Fu and Fu 1996, Cheung, Han, Ng, Wong 1996, Ayad, El-Makky and Taha 2001, Cheng, Yan, and Han 2004, Ayan, Tansel and Arkun 1999, Chang, Yang 2003, Cheung, Lee and Kao 1997) have been developed and are mostly focused on minimizing the number of database rescanning.

Copyright (c) 2007, Australian Computer Society, Inc. This paper appeared at the Eighteenth Australasian Database Conference (ADC2007), Ballarat, Victoria, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 63. James Bailey and Alan Fekete, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

Currently there is no updating algorithm available for temporal association rules. In our work, we have developed a new algorithm call *ITAR* for incremental mining of temporal association rules by employing the skeleton of *ICAP* (Ayad, El-Makky and Taha 2001). For the rest of the paper, section 2 provides a review of related works *ITAR*. Section 3 gives a description of the incremental mining problem of temporal association rules. Section 4 presents the proposed algorithm *ITAR* in details. The section afterwards describes how the algorithm has been adopted for crime pattern discovery. Another real life dataset has been used to experiment with the new algorithm and the results are presented in section 6. Section 7 concludes our work.

2 Related Works

2.1 Temporal Association Rules with Numerical Attributes

The definition of temporal association rules is provided in the works of Wang, Yang and Muntz (1999, 2001) together with the introduction of the TAR (Temporal Association Rule) algorithm. It deals with objects in a database, *DB*. Each object O_i has a unique ID and a set of numerical attributes A_1, A_2, \dots, A_n . The *DB* is viewed as a set of sequences of snapshots S_1, S_2, \dots, S_t and each snapshot has objects consisting of attributes with their numerical values.

Evolutions are defined to represent the temporal changes of attribute value of some object histories. Given an attribute A_i , an attribute evolution $E(A_i)$ of length m describes the value change of A_i over m consecutive snapshots. $E(A_i) = (A_i ? [l_1, u_1]) ? (A_i ? [l_2, u_2]) ? \dots ? (A_i ? [l_m, u_m])$ represents the changes of A_i over m consecutive snapshots where $[l_j, u_j]$ is the interval of values of A_i , and is the union of a sequence of adjacent base intervals. For example, an employee's salary changes from an interval of \$40,000 and \$45,000 to an interval of \$50,000 to \$55,000, and then to an interval \$60,000 to \$65,000. This example represents an evolution that across 3 snapshots regardless of it is consecutive or not. The attribute value is presented in the interval form instead of distinct values.

A temporal association rule can be discovered from base cubes as long as it achieves the minimum *support* and *strength*. Let A_1, A_2, \dots, A_n be a subset of attributes. A temporal association rule, R , of length m is defined as:

- $E(A_1) \ E(A_2) \ \dots \ E(A_{k-1}) \ E(A_{k+1}) \ \dots \ E(A_n) \Rightarrow E(A_k)$, and
- $E(A_i)$ is an evolution of length m with attributes A_i ($1 \leq i \leq n$).

In TAR, three thresholds, which are support, density and strength, are introduced to select interesting temporal association rules. The support of a temporal rule $E(A_1) \cup E(A_2) \cup \dots \cup E(A_{k-1}) \cup E(A_{k+1}) \cup \dots \cup E(A_n) \Rightarrow E(A_k)$ is the support of an evolution conjunction $E(A_1) \cup E(A_2) \cup \dots \cup E(A_n)$, that is the number of object histories of length m which follow this evolution conjunction. The density is the ratio of the number of objects to total number of objects, which will make a base cube considered “dense”. The base cube is defined as evolution cube which consist of a set of attributes and whose values fall into base interval regarded as non-distinguishable. The density is introduced to ensure that a cube will not be included as part of a rule if there is no enough evidence to show that the rule holds for the cube. The strength of an association rule is $\text{Support}(E(A_1) \cup E(A_2) \cup \dots \cup E(A_n)) / \text{Support}(E(A_1) \cup E(A_2) \cup \dots \cup E(A_{k-1}) \cup E(A_{k+1}) \cup \dots \cup E(A_n))$. The strength indicates the degree of dependence between attributes (A_1, A_2, \dots, A_n) of the rule.

The TAR algorithm has two phases. The first phase is to find all base cubes, *BaseCube*, which satisfy the density threshold after partitioning each attribute domain into b base intervals. Based on dense base cubes, the second phase finds all valid temporal rule sets. In our work, we focus on refining phase 1. It is because updating original database by appending incremental database affects the densities and the distributions of existing discovered base cubes, i.e. *BaseCube*.

2.3 Incremental Mining of Association Rules

The problem of incremental mining was solved by the FUP algorithm (Cheung, Han, Ng, Fu and Fu 1996). Let L^{DB} be a set of frequent itemsets in a transactional database DB . Let L^{db} be a set of frequent itemsets in an incremental database db that is going to be appended to DB then eventually form $DB+$. With the same support threshold, it is required to discover L^{DB+} . The output of L^{DB+} is a new set of frequent itemsets as the result of incremental mining. The itemsets of winners and losers are defined below.

- **Winners:** infrequent itemsets of DB that become frequent in $DB+$ after adding the increment data to the database.
- **Losers:** frequent itemsets of DB that become infrequent $DB+$ after adding the increment data to the database.

Let X be an itemset and superscript DB or db present the type of database. In order to discover the winners and losers from itemsets, itemsets can be classified into four types from L^{DB} and L^{db} as shown in Figure 1.

	$X \in L^{db}$	$X \notin L^{db}$
$X \in L^{DB}$	1	2
$X \notin L^{DB}$	3	4

Figure 1. Itemset updating scenarios.

Scenarios (1) and (4) in Figure 1 are easy to handle because itemsets belong either (1) and (4) represent itemsets definitely become winners and losers, respectively. Itemsets in scenario (2) and scenario (3) are the uncertainty areas because itemsets can become either winners and losers. Scenario (2) would be the easier one because support counts of itemsets in L^{DB} are already available and itemsets can become winners as long as updated support counts are equal or exceed the minimum support. Scenario (3) would require rescanning the original database since the support counts of itemsets in original database DB were unknown.

The major drawback of the *FUP* is that it was built on the *Apriori* algorithm with small modifications. Many passes of checking against the original database would usually be required. To remedy this problem, a concept of negative border introduced by Toivonen in 1996 can indicate the need for an update of rules. Informally, the negative border of a set of frequent itemsets L consists of all the itemsets that are not in L but have all their subsets in L . The itemsets in negative border for subsequent incremental mining have potential to become a winner as long as it achieves the minimum support threshold. On the other hand, if no winner appeared in the negative border, there is no need for checking further in the original database. The negative border, therefore, can provide a signal of the need for updating the database. It has been also used in *ICAP* (Ayad, El-Makky and Taha 2001) and *IUS* (Zheng, Xu, and Ma 2002) to improve the efficiency of speed and minimize the use of resources during mining.

3 Incremental Mining of Temporal Association Rules

3.1 Incremental Mining Problem

The problem of incremental mining of temporal association rules is mostly identical to the one in general association rules. The idea is to discover new sets of dense base cubes by reallocating the average density in hypercubes of evolution spaces when there is another set of objects db added into DB .

$$\text{Let } Dense^{DB} = \bigcup_{i=1}^m \bigcup_{j=1}^T \text{BaseCube}(i, j) = \{bc \mid bc \geq \text{density}\}$$

be a set of dense base cubes from all sliding windows of DB that satisfies the density threshold where m is width of window and T is the number of attributes. An incremental database db can be added to DB and the $Dense^{DB}$ is needed to be updated.

After updating, all winners are updated to $Dense^{DB+}$ and losers are updated to temporal negative border $TNBd$. The negative border, $TNBd$, keeps all losers that were the

winners in previous DB . The concept of temporal negative border here is dissimilar to original negative border. It aims to avoid generating un-necessary powersets of base cube from the winners. Further details of temporal negative border and its comparison will be presented in Section 3.3. Before continuing our discussion, the notations used in this paper are shown in Figure 2.

DB	The original database
db	The incremental database
$DB+$	The updated database ($DB+ = DB \cup db$)
$ DB , db , DB+ $	Number of objects in $DB, db,$ and $DB+$, respectively
T^{DB}, T^{db}, T^{DB+}	Set of snapshots in $DB, db,$ and $DB+$, respectively
$T_s^{DB}, T_e^{DB}, T_s^{db}, T_e^{db}, T_s^{DB+}, T_e^{DB+}$	Starting snapshot and ending snapshot in $DB, db,$ and $DB+$.
O^{DB}, O^{db}, O^{DB+}	Set of objects in $DB, db,$ and $DB+$, respectively
bc	A base cube of the set of attribute evolution follows the same temporal sequence
$Dense^{DB}, Dense^{db}, Dense^{DB+}$	Dense base cubes for $DB, db,$ and $DB+$, respectively
$BaseCube(i, m)$	Level-wise regional base cube having i distinct attributes and length m of evolutions
db^n	n^{th} incremental database for mining
m	Size of sliding window
b	Number of intervals for attributes
min_sup	Minimum support threshold
e	Density threshold
$W(j, m)$	Sliding window with width m and starting snapshot is j

Figure 2: Notations used.

3.2 Different Cases in Updating Temporal Association Rules

We adopt the similar approaches to handle INSERT and APPEND cases as in *IncSpan* (Cheng, Yan, and Han 2004). INSERT case represents pure object growth in a database regardless to snapshot domain coverage. New objects in db might introduce new timestamps (snapshots) of attributes. Also, db might introduce itemsets with new atomic items which were not existed in DB . The APPEND case presented in *IncSpan* adds itemsets at the end of sequence of an existing sequence, where $T > T_e^{DB}$. In terms of snapshot coverage, APPEND case can also include previous snapshots, where $T_s^{DB} \leq T \leq T_e^{DB}$ and $T \notin T^{DB}$ and $T < T_s^{DB}$ and $T \notin T^{DB}$. After updating, T_s^{DB+} and T_e^{DB+} will be updated. The coverage of the db in DB could be classified into 3 types of coverage, total overlapping, partial overlapping and non-overlapping.

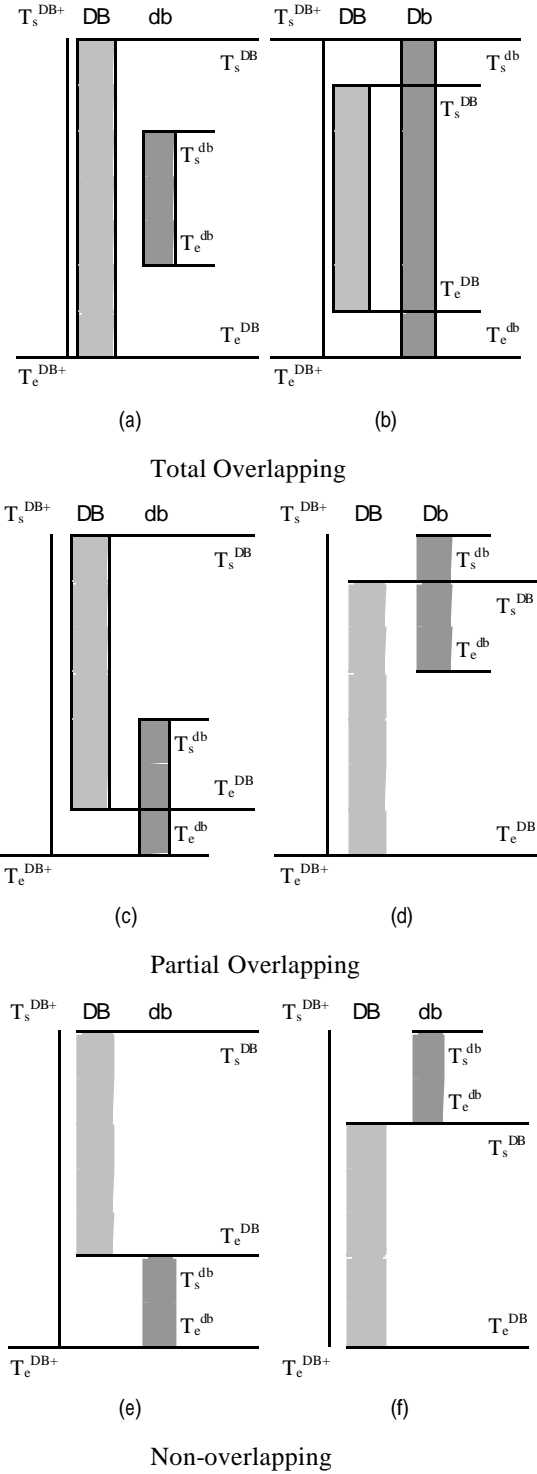


Figure 3: Snapshot Coverage for APPEND

Total overlapping is that the domain of snapshots of either DB or db fully covers each other. Case (a) in Figure 3 does not have any new snapshot to be introduced into DB , where $T^{db} \subseteq T^{DB}$, and $T_s^{DB+} = T_s^{db} = T_e^{DB+} = T_e^{db}$. Case (b) in Figure 3 introduces new snapshots which beyond both sides of domain, where $T^{DB} \subseteq T^{db}$, and $T_s^{db} = T_s^{DB+} = T_e^{DB+} = T_e^{DB}$. The domain of snapshots in $DB+$ will be updated to same as db .

Partial overlapping is that some snapshots of db cover the domain snapshots in DB . Case (c) in Figure 3 represents that db introduces new subsequent snapshots at the end point of domain where $T_s^{DB+} = T_s^{db} = T_e^{DB+} = T_e^{db}$ and the

updated domain will be $T_s^{DB+} = T_s^{DB}$, $T_e^{DB+} = T_e^{db}$ in $DB+$. Case (d) in Figure 3 has new snapshots from db complementing aforesaid side where, $T_s^{db} = T_s^{DB} = T_e^{db} = T_e^{DB}$, and the updated domain will be $T_s^{DB+} = T_s^{db}$, $T_e^{DB+} = T_e^{DB}$.

Non-overlapping is that there is no overlapped snapshot in db . Case (e) in Figure 3 is similar to the APPEND case in *IncSpan* where new subsequent snapshots attributes are to be introduced and $T_s^{DB} = T_e^{DB} < T_s^{db} = T_e^{db}$ and updated domain for $DB+$ will be $T_s^{DB+} = T_s^{DB}$, $T_e^{DB+} = T_e^{db}$. Case (f) in Figure 3 is the inverse version of case (e) where $T_s^{db} = T_e^{db} < T_s^{DB} = T_e^{DB}$, and updated domain for $DB+$ will be $T_s^{DB+} = T_s^{db}$, $T_e^{DB+} = T_e^{DB}$.

Besides timestamp considerations, new objects may be added during updating. There can be five different cases of temporal database insertion. In order for better explanation, we illustrate with a sample database. Given a database with one object, and let $I = \{A_1, A_2, A_3, \dots, A_6\}$ be its set of literals. The object consists of set of items with timestamp as $\{\{T_1: A_1, A_2, A_3\}, \{T_2: A_2, A_3, A_5\}, \{T_3: A_1, A_3, A_5, A_6\}, \{T_4: A_1, A_2, A_3\}\}$. The object could be transformed to be a set of atomic items $\{A_1T_1, A_2T_1, A_3T_1, A_2T_2, A_3T_2, A_5T_2, A_1T_3, A_3T_3, \dots, A_3T_4\}$. Therefore, the set of atom items will be extended if the timestamp of new data is growing.

- *Case 1: Updating values of existing atom items:* Atomic items are modified for an existing object. For example, atomic items $A_1T_1=2$ and $A_2T_2=4$ are to be replaced by $A_1T_1=3$ and $A_2T_2=5$, where $T_s^{DB} < T < T_e^{DB}$. After updating, the average density of base cubes might be affected because the new values of relevant items might be fall into other interval.
- *Case 2: Inserting items into existing snapshots:* In this case, un-existed items $X ? I$ are to be supplemented. For example, items A_4T_1 , A_2T_3 , and A_4T_3 with their values, which were not existed in DB and where $A_i ? I$ and $T_s^{DB} < T < T_e^{DB}$, are intersect into the existing histories. Therefore, the object is updated to $\{A_1T_1, A_2T_1, A_3T_1, A_4T_1, A_2T_2, A_3T_2, A_5T_2, A_1T_3, A_2T_3, A_3T_3, \dots, A_3T_4\}$. After updating, the average density of base cubes would be affected because new atomic items were inserted into DB .
- *Case 3: Appending subsequent snapshots:* Introduces new timestamp items where $A_i ? I$ and $T > T_e^{DB}$. For example, $\{\{T_5: A_2, A_3, A_4\}, \{T_6: A_4, A_6\}\}$ insert into subsequent object history.
- *Case 4: Inserting new items into existing snapshots:* From case 1 to case 3, the items are only the member of existing literals. However, db can introduce some new literals such as in set $J = \{A_7, A_8, \dots, A_{10}\}$, where $J \cap I = \emptyset$. This is similar to case 2 but incremental data has new literals. Incremental database db consists of $\{\{T_7: A_7, A_8\}, \{T_8: A_8, A_9\}\}$, where $T_s^{DB} < T < T_e^{DB}$ and $A_i ? J$.
- *Case 5: Appending subsequent snapshots with new literals of item:* This is similar to case 3 but

incremental data is new literals of item. Incremental database db consist of $\{\{T_5: A_7, A_8\}, \{T_8: A_8, A_9\}\}$, where $T_n > T_e$ and $A_i ? J$.

In our work, we focus on case 3, case 4 and case 5 in which new objects are inserted or appended in the incremental database.

3.3 Temporal Negative Border

Since mining temporal association rules is restricted by sliding windows and temporal sequence, the original concept of negative border when applied directly would introduce the following problems:

- The power set, $P(R)$ where R is a temporal association rule, generation approach by the original negative border for the base cubes is not practical because the set of literals for obtaining atomic items in base cubes can become very large since temporal evolution is continuous in nature.
- Constructing level-wise dense base cube *Dense* of every slide window lattice employs the *AprioriGen* method. However, it requires additional computation effort to construct the set of minimal itemsets $X ? R$ not in *Dense*. Computation effort includes time for processing, and memory and I/O usage for generating and storing all power set of rules $P(R)$ from frequent itemsets to negative border by the *AprioriGen* method.
- The updating problem of temporal rules has a fundamental difference from generic association rules in which the set of literals $I = \{i_1, i_2, \dots, i_3\}$ was well established and predictable. In temporal association rule mining, snapshot evolution is continuously growing and the set of literals becomes large. Consequently, the size of negative border becomes relatively large.

Hence, we define a variation of negative border, called the *temporal negative borde (TNB)r*, to simplify the computation problem above. The TNB of set of base cubes, $Dense^{DBn}$ which follow the past object histories in evolution cubes, is referred to as $TNBd(Dense)$. That is $TNBd(Dense^{DBn}) = \{bc \mid bc \notin Dense^{DBn} \text{ and } bc ?$

$\bigcup_{i=1}^{n-1} Dense^{dbi}$, where $1 \leq i < n$. In other words, it says to

include all the losers in the last incremental mining if these base cubes have been winners before.

Suppose DB is the initial database and subsequent incremental database are represented as db^i , where $1 \leq i \leq n$. The negative border in the first mining is empty since there was no existing negative border. From second mining iteration onward, the negative border would keep previous losers. There are 3 sets of dense bases, $Dense^{DB}$, $Dense^{db}$, and $TNBd(Dense^{DB})$ to be processed for each mining iteration. The $TNBd(Dense^{DB})$ in second mining is still empty. $TNBd(Dense^{DB+})$ is formed from the losers of DB . Since the losers from all $Dense^{db}$ are never to be considered, for any base cube, bc , in $Dense^{DB}$, it must be winners of $Dense^{db}$. Therefore, the losers from $Dense^{DB}$

must also be the winners of some previous db . There are several benefits in using the temporal negative border:

- It is able to eliminate un-necessary sets of base cubes. It would not generate powersets that do not exist in db nor DB .
- With using the new negative border, the rescanning of the complete database can be at most one time. The new base cubes having new snapshot(s) or existing snapshot(s) not occurred in previous $Dense$ are discovered.
- It handles the problem of the INSERT and APPEND cases. In temporal association rules, the INSERT case introduces object histories of new objects. It can be easily handled based on the property of support counts between winners and losers as well as the intersections mapping concept in *ICAP* (Ayad, El-Makky and Taha 2001, Cheng). The APPEND case involves new discovered base cubes in $Dense^{db}$, where base cubes were not previously discovered both in $Dense^{DB}$ and $TNBd$. Attached snapshots in such base cubes are either new ($T \geq T_e^{DB}$ or $T \leq T_s^{DB}$) or existing timestamps ($T_s^{DB} \leq T \leq T_e^{DB}$). New discovered base cube will be required an original database rescan. In our algorithm, it can deal with the INSERT case and the hybrid of INSERT and APPEND case since db only contains new objects.

4 Incremental TAR Algorithm

Our Incremental Temporal Association Rule (*ITAR*) algorithm as shown in Figure 4 has two assumptions. The first one is the incremental database db only contains new objects with their details which were not existed in DB . Second, if a base cube bc in db satisfied the density threshold before the database is updated, bc will satisfy the density threshold in $DB+$ as well; otherwise, bc is never to be considered. No matter bc was already in either $Dense^{DB}$ or $TNBd(Dense^{DB})$.

The *ITAR* algorithm takes input as the set of dense base cubes $Dense^{DB}$ of the original database DB and the negative border $TNBd(Dense^{DB})$. Incremental database db is to be mined separately using *TAR* algorithm that delivers a new set of dense base cubes $Dense^{db}$. After the merging of three sets of dense base cubes, five different groups from the sets can be identified and each group will be respectively processed. The proposed algorithm can be summarized in the following steps:

1. Use the original *TAR* algorithm to discover dense base cubes (all level-wise base cubes of all sliding window) denoted as $Dense^{db}$ in the incremental database db .
2. With the discovered $Dense^{db}$, all intersections of $Dense^{db}$, $Dense^{DB}$, and $TNBd(Dense^{DB})$ could be identified. After accumulating each base cube's density, winners will be fall into $Dense^{DB+}$ meanwhile losers will be fall into new negative border, $TNBd(Dense^{DB+})$.

3. After step 2, the remaining bases cubes would be filtered from the sets of intersections. Remaining base cubes in $Dense^{DB}$ are required to check their densities again for the updated object counts. On the other hand, remaining base cubes in $TNBd(Dense^{DB})$ will automatically be fall into $TNBd(Dense^{DB+})$ since such base cubes are not the winners in current mining.
4. All base cubes of $Dense^{DB}$ and $TNBd(Dense^{DB})$ have been offset. The remainders in $Dense^{db}$ represent those were not occurred in both $Dense^{DB}$ and $TNBd(Dense^{DB})$ but are new discovered base cubes from db . It requires rescanning the original database to updated densities for remainders. If a base cube bc satisfies the density threshold, bc will fall into $Dense^{DB+}$; otherwise, it will fall into $TNBd(Dense^{DB+})$.

ITAR Algorithm:

Function *ITAR* ($Dense^{DB}$, $TNBd(Dense^{DB})$, db)

1. Compute $Dense^{db}$ using *TAR* algorithm
2. $Dense^{DB+} = \emptyset$; $TNBd(Dense^{DB+}) = \emptyset$
3. $|DB+| = |DB| + |db|$ // Accumulate the number of object to $DB+$
4. **For each** base cube bc in $Dense^{DB}$ $Dense^{db}$ **do** // (1)
 5. $bc.count$ of $DB+ = bc.count$ of $DB + bc.count$ of db
 6. $Dense^{DB+} = Dense^{DB+} \cup \{bc\}$
7. **For each** base cube bc in $TNBd(Dense^{DB})$ $Dense^{db}$ **do** // (2)
 8. $bc.count$ of $DB+ = bc.count$ of $DB + bc.count$ of db
 9. **if** $bc.count$ of $DB+ \geq |DB+| / b \times e$ **then**
 10. $Dense^{DB+} = Dense^{DB+} \cup \{bc\}$
 11. **else** $TNBd(Dense^{DB+}) = TNBd(Dense^{DB+}) \cup \{bc\}$
12. **For each** base cube bc in $Dense^{db}$ **do** // (3)
 13. **if** $bc.count$ of $DB+ \geq |DB+| / b \times e$ **then**
 14. $Dense^{DB+} = Dense^{DB+} \cup \{bc\}$
 15. **else** $TNBd(Dense^{DB+}) = TNBd(Dense^{DB+}) \cup \{bc\}$
17. **For each** base cube bc in $TNBd(Dense^{DB}) \{bc \mid bc \notin Dense^{db}\}$ **do** // (4)
 18. $TNBd(Dense^{DB+}) = TNBd(Dense^{DB+}) \cup \{bc\}$
19. **if** $Dense^{db} \{bc \mid bc \notin TNBd(Dense^{DB})$ and $bc \notin Dense^{DB+}\} = \emptyset$ **then** // (5)
 20. **For each** O in DB **do**
 21. **For all** cube bases bc in O **do**
 22. **if** $bc \in Dense^{db}$ **then**
 23. $bc.count$ of $DB+ = bc.count$ of $DB+ ++$
 24. **For each** bc in $Dense^{db}$ **do**
 25. **if** $bc.count \geq |DB+| / b \times e$ **then**
 26. $Dense^{DB+} = Dense^{DB+} \cup \{bc\}$
 27. **else**
 28. $TNBd(Dense^{DB+}) = TNBd(Dense^{DB+}) \cup \{bc\}$
 29. **Return** $Dense^{DB+}$ and $TNBd(Dense^{DB+})$

Figure 4: A high level description of algorithm *ITAR*

The idea of *ITAR* is to merge the dense base cubes between the intersections of $Dense^{DB}$, $Dense^{db}$, and $TNBd(Dense^{DB})$. During step 4, rescanning the original database is required if and only if the remaining base cubes do not belong to any intersections from *db*.

The original negative border (Toivonen 1996) made use of the idea in the *Apriori* algorithm to generate all candidate itemsets where negative border of *L* consists of minimal itemsets *X* not in *L*. Our variation of negative border does not possess the same property. The proposed temporal negative border avoids the power set of base cube generation. A smaller negative border is exploited to improve mining performance by cutting down redundant or insignificant base cube checking. For example, suppose a base cube $bc = \{E(A), E(B), E(C)\}$ was never a winner previously but subsets ($\{E(A)\}$, $\{E(B)\}$, $\{E(C)\}$, $\{E(A), E(B)\}$, ..., $\{E(B), E(C)\}$) of *bc* were the winners. The original negative border would hold *bc* which might be already by power set. However, temporal negative border never keeps it.

5. Crime Pattern Discoveries

We have adopted the new algorithm, *ITAR*, to support a crime pattern analysis for a district in Hong Kong for better computational performance. In the developed system, an end-user can make a request for mining of the temporal association rules through the interface of the system. The interface will accept a user request and then the corresponding query is generated for collecting corresponding crime data from the database. The query specifies the day range (week, month), attributes and grid size, and counts of crime incidence to be utilized. The query can also filter crime data with missing values. In order to generalize the coordinates of crime data, coordinates are divided into grids for better spatial mining results. Figure 11 shows a system architecture of the crime pattern analysis system.

Crime information extraction is done by *crime extractor*, which collects and analyses user requests, collects corresponding crime data from the database, processes the data and then passes the data to *dense base cube generator* to create base cubes for mining. The results of the *dense base cube generator* is used for examining crime trends, generating spatial-temporal associations, or mining incrementally. The mining results are returned to the end-user through a graphical interface. The back-end database stores the crime data for analysis. Dense base cubes generated from the *dense base cube generator* are also stored in the database to prepare for incremental mining. When the system requires mining incrementally, the original dense base cubes can be retrieved from the database.

Crime trend discovery is done by *crime trend generator*, which collects dense base cubes generated from *dense base cube generator*, analyses snapshots in dense base cubes and outputs the trend results. In order to generate a dense base cube, the density of the conjunction of snapshots in the cube must be greater than the density threshold. The density thresholds can control the generation of trend, so that if there are enough records or

evidence to show that a trend exists, the trend will be discovered.

Incremental base cube discovery is done by *incremental base cube generator*, which collects original dense base cubes generated from *dense base cube generator* or the database, mines dense base cubes from incremental database, merges the dense base cubes of incremental database to cubes of original database and passes the updated dense base cubes to the *spatial-temporal association rules generator* to generate the rules.

Since continuous temporal change of value of crime attributes is mined, the crime attributes must be in numerical form. However, there is no numerical attribute, except coordinates, in the crime data provided. The offence and mo (modus operandi, which means committing crime method) attributes of crime data are categorized into different levels of seriousness, with values from 1 to 10. Figure 5 and Figure 6 shows some examples of categorizing of offence and mo respectively.

Offence	Seriousness
Affray	1
Breach of condition of stay	2
Resisting arrest	3
Lending at excessive rate	4
Criminal damage	5
Robbery, deception	6
Burglary	7
Criminal intimidation	8
Assault	9
Arson, child abuse	10

Figure 5. Examples of Categorizing of Offence

MO	Seriousness
Loitering – causing others to be concerned	1
Loitering – with intent to commit an arrestable offence	2
Theft – steal from locker	3
Theft – pickpocket – cut open	4
Deception – modelling school	5
Robber – use knife, fruit knife 20cm long	6
Wounding – minor dispute	7
Possession of dangerous drug –stop and search	8
Possession of dangerous drug -operation	9
False imprisonment – escort back and detain to settle debt	10

Figure 6. Examples of Categorizing of MO

Although the crime data is stored in a transaction database, it can be viewed as a sequence database, which records a set of snapshots that capture crimes happened at different locations in different time intervals (week or month). Each base snapshot generated from crime data records the values of a set of crime attributes in base intervals. For example, a base snapshot may record that there are ten crimes with offence=4 happened at grid (x=4, y=8) in January. When association relationship (crime1 \Rightarrow crime2) between two crimes happened in different locations are tried to discover, base snapshots must record attributes of the two crimes. However, there is no single crime record consists of two crimes happened at two different locations, so single crime record cannot be used to generate base snapshot with two crimes. In order to grouping two crime records together to generate the base snapshot, we assume that when two crimes happened in the same time interval (week or month), there is correlation relationship between the crimes. For example, a base snapshot may record that there are ten crime incidence happened in January, one of the crimes with offence=4 happened at grid (x=4, y=8), another crimes with offence=8 happened at grid (x=10, y=14).

The spatial-temporal crime analysis system has been developed by the JAVA 2 programming language (JDK 1.5.0_06). Eclipse 3.0 Software Developer's Kit was used as development platform for debugging and building the system. MySQL server 5.0 was used as database of the system. Figure 7 shows the interface of the system.

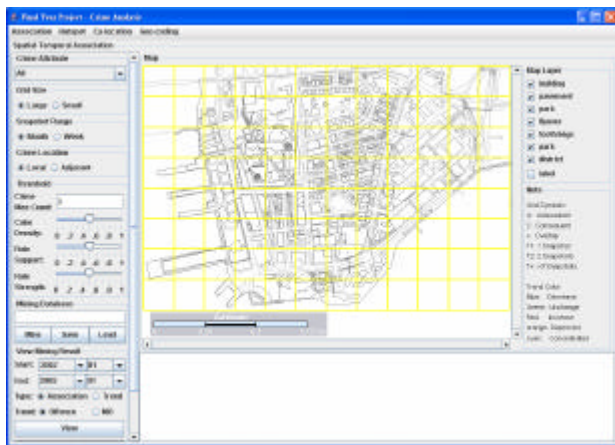


Figure 7. Crime Pattern Analysis System

6 Experiments

6.1 Environment and Configuration

For the verification of the performance of the ITAR algorithm, we have tried it with a real data set. The data used here is a transactional database with customers shipment details for the whole year of 2002 obtained from a shipping company. There are a total of 40,000 customer transactions to represent the daily histories of shipment services used. Each transaction has a customer ID, and a shipment product with its total volume of the same day. The snapshot was taken once a month, from Jan 2002 to Dec 2002. With the real dataset, a synthetic temporal database for the experiments is generated by means of

SQL with grouping and aggregation techniques. Each element of the database contains an object with its identifier and object histories of 12 snapshots (months). The properties of the synthetic temporal database are summarized and listed in Figure 8.

Number of objects in temporal database	40,000
Number of attributes	5
Period	1 Jan 2002 to 31 Dec 2002
Snapshots	Each month represent 1 snapshot

Figure 8. Synthetic dataset properties for experiments

Furthermore, the synthetic temporal database with 40,000 objects has also been randomly partitioned into 20 sets of small and equal size datasets as incremental databases. Each dataset consists of 2,000 objects (5 percents of the original database) with their object histories. Meanwhile, we used these small datasets to generate another 20 sets of accumulated datasets for different mining iterations. For example, the accumulated dataset for n^{th} iteration incremental mining has the 1st to n^{th} small datasets. There are two types of experiments. The first type is to compare the mining speed between ITAR and TAR. The second type is to verify the resultant temporal rules generated by the two algorithms respectively.

6.2 Mining Performance

In executing ITAR, the number of base intervals for attributes in the dataset, the width of sliding window, and the density threshold are the important parameters that can determine required runtime.

We divided the set of experiments that used different parameters: attribute intervals from 10 to 40, the widths of sliding windows from 3 to 5, and density thresholds from 0.5 to 2. In each experiment, we fixed the values for two parameters and adjusted the remaining one. The Accumulated ITAR (AITAR) algorithm is implemented as the TAR algorithm was applied to the complete database every time when there is an update on the database. Figure 9 shows the results of ITAR algorithm under different setups of parameters. In Figure 10, it shows the ratio of mining efficiency between ITAR and TAR. Second, it shows the ratio of mining efficiency between accumulated ITAR (AITAR) and TAR.

In both Figures 9 and 10, the performance of ITAR with different parameters is very stable and showing good performance. In contrast, the performance of TAR shows a linearly growth along the mining sequence.

7 Conclusion

Researches in temporal association rules mining have been developing for many years. Due to the re-scanning issue in TAR in order to handle dataset updating, an incremental

algorithm *ITAR* (Incremental *TAR*) is proposed in this paper. The algorithm applies the techniques of incremental mining on the *TAR* to update the dense base cubes after the update of the original database by adding new objects with temporal histories. Due to the nature of the temporal association rules mining model, a variation of negative border is proposed for *ITAR* in order to simplify the use of the original. Temporal negative border proposed in this paper only keeps all past winners which are becoming losers instead of comprising the power set of dense base cubes. Therefore, the number of losers of base cubes held by temporal negative border can be minimized. The *ITAR* algorithm has been adopted in a crime pattern analysis system. Furthermore, two sets of experiments have been conducted to measure the relative performance of the new algorithm compared to the *TAR* algorithm. The results reflected that the exhibitions of *ITAR* benefit a significant efficiency against *TAR* in term of efficiency ratio. Moreover, using temporal negative border enables the avoidance of any missing base cube when using *ITAR*.

Acknowledgement

The work of the authors is supported in part by the CERG RGC Grant of research project code PolyU 5106/05E (BQ938).

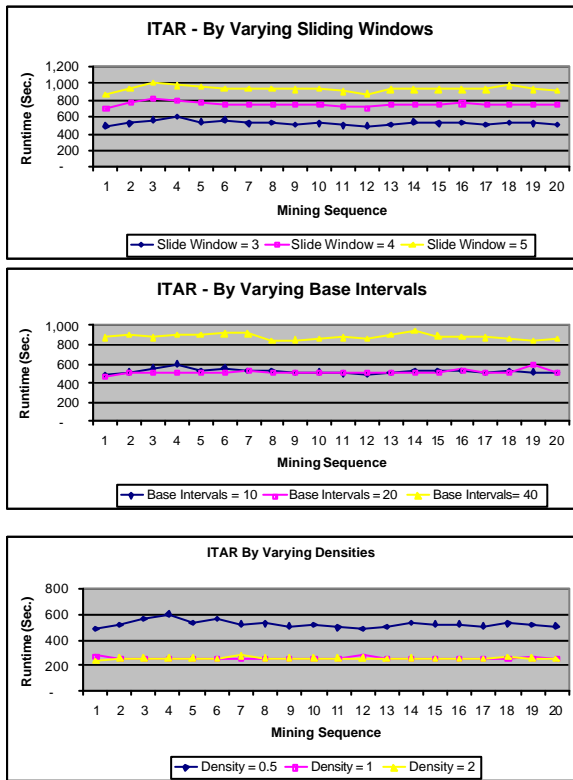
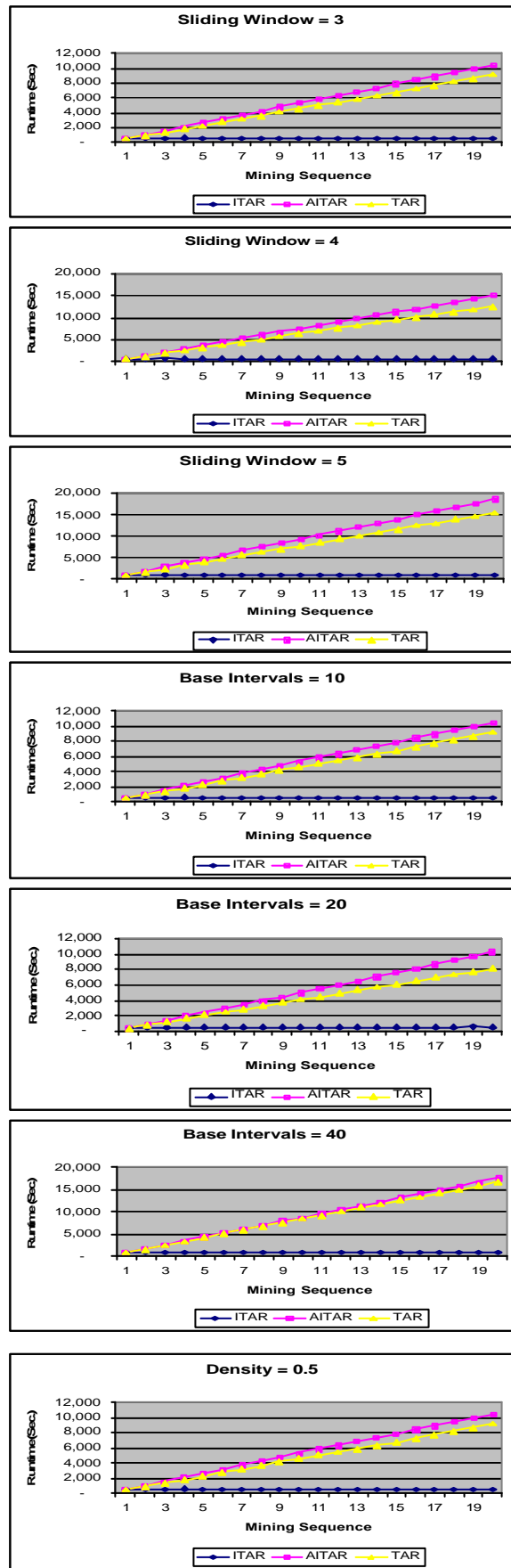


Figure 9. Mining Efficiency of *ITAR* comparison by different parameters



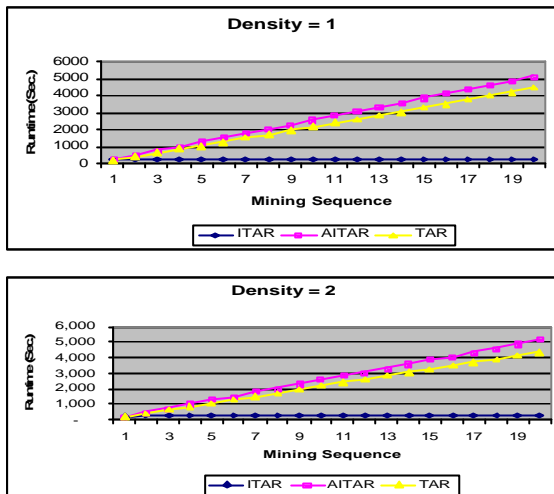


Figure 10. Mining Efficiency comparisons (by Varying Density thresholds)

8 References

- Agrawal, R., Imielinski, T. and Swami, A. (1993): Mining association rules between sets of items in large databases. *Proc. ACM SIGMOD International Conference on Management of Data*, Washington DC, USA, **22**:207-216, ACM Press.
- Agrawal, R., and Srikant, R. (1994) Fast algorithms for mining association rules. *Proc. of the 20th Int'l Conf. on Very Large Database*, Santiago, Chile, Sept. 1994, pp. 487-499.
- Ayad, A., El-Makky, N., and Taha, Y. (2001) Incremental Mining of Constrained Association Rules. In *First International SIAM Conference on Data Mining (SDM01)*, Chicago, April 2001. http://www.siam.org/meetings/sdm01/pdf/sdm01_01.pdf.
- Ayan, N.F., Tansel, A.U., and Arkun, E. (1999) An efficient algorithm to update large itemsets with early pruning. *The 5th Conference on Knowledge Discovery and Data Mining (SIGKDD99)*, AMC Press, San Diego, USA, pp. 439-450.
- Cheng, H., Yan, X., and Han, J. (2004) "IncSpan: Incremental Mining of Sequential Patterns in Large Database", *Proc. 2004 Int. Conf. on Knowledge Discovery and Data Mining (KDD'04)*, Seattle, WA, Aug. 2004, pp. 527-532.
- Chang, C.H., Yang S.H. (2003) Enhancing SWF for incremental association mining by itemset maintenance. *The 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining, PKADD 203, LNAI 2637*, Springer, Seoul, Korea, pp 301-312.
- Cheung, D.W., Han, J., Ng, V., Fu, A., and Fu, Y. (1996) A fast distributed algorithm for mining association rules. In *Proc. 1996 Int. Conf. Parallel and Distributed Information Systems*, pp. 31-44, Miami Beach, FL. Dec 1996
- Cheung, D.W., Han, J., Ng, V., and Wong, Y. (1996) Maintenance of discovered association rules in large databases: An incremental updating technique. In *Proc. 1996 Int. Conf. Data Engineering (ICDE'96)*, pp. 106-114, New Orleans, LA, Feb. 1996
- Cheung, D.W., Lee, S.D., Kao, B. (1997) A general incremental technique for maintaining discovered association rules. *The 5th International Conference on Database Systems for Advanced Applications*, Melbourne, Australia, pp. 185-194.
- Toivonen, H. (1996) Sampling large database for association rules. In *22nd International Conference on Very Large Database (VLDB'96)*, pp. 135-145, Mumbai, India, September 1996.
- Wang, W., Yang, Y., and Muntz, R. (1999) *Temporal Association Rules with Numerical Attributes*. NCLA CSD Technical Report 990011, 1999.
- Wang, W., Yang, Y., and Muntz, R. (2001) TAR: temporal association rules on evolving numerical attributes., *Proceedings of the 17th IEEE International Conference on Data Engineering*, pp. 283-292, 2001.
- Zheng, Q., Xu, K., and Ma, S. (2002) *The Algorithm of Updating Sequential Pattern*, 2002, <http://xxx.arxiv.org/ftp/cs/papers/0203/0203027.pdf>.

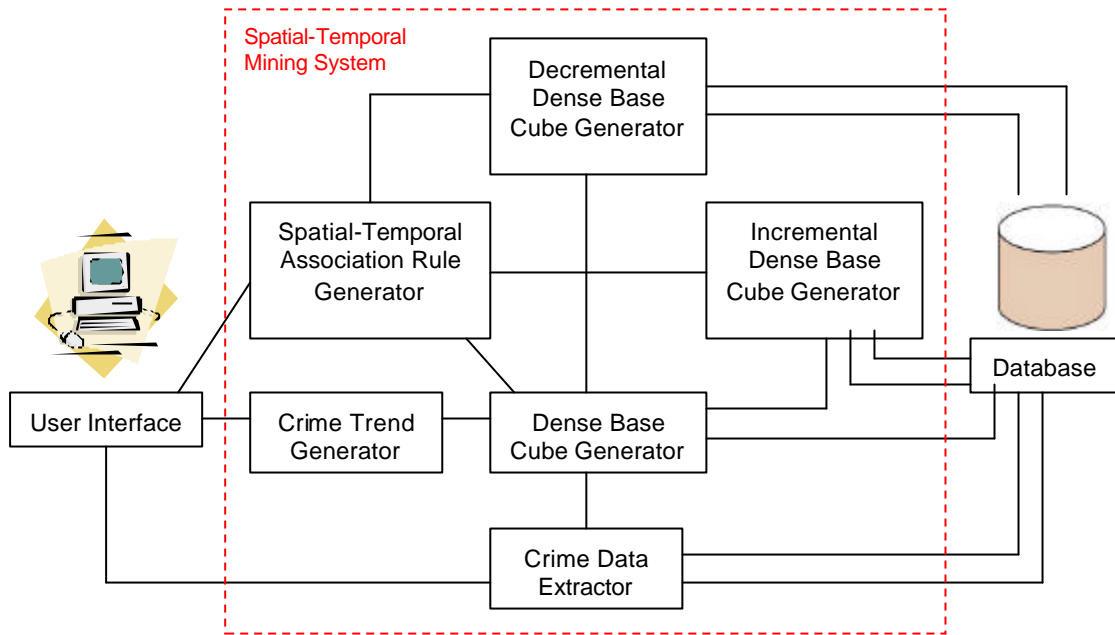


Figure 11. Overview of the Crime Pattern Analysis System