International Journal of Computer Science and Informatics

Volume 1 | Issue 4 Article 7

April 2012

A Mark Based Indexing Method for Well organized Data Retrieval of Comparative Temporal Patterns

Swamy Babu Nidamanuri
CSE, QISCET, JNTU-Kakinada, Ongole, India, nidamanuri.swamybabu@gmail.com

R. Lakshmi Tulasi
CSE, QISCET, JNTU-Kakinada, Ongole, India, ganta.tulasi@gmail.com

Recommended Citation
Nidamanuri, Swamy Babu and Tulasi, R. Lakshmi (2012) "A Mark Based Indexing Method for Well organized Data Retrieval of Comparative Temporal Patterns," *International Journal of Computer Science and Informatics*: Vol. 1 : Iss. 4 , Article 7.
DOI: 10.47893/IJCSI.2012.1049
Available at: https://www.interscience.in/ijcsi/vol1/iss4/7

A Mark Based Indexing Method for Well organized Data Retrieval of Comparative Temporal Patterns

Swamy Babu Nidamanuri & R. Lakshmi Tulasi
CSE, QISCET, JNTU-Kakinada, Ongole, India
E-mail : nidamanuri.swamybabu@gmail.com, ganta.tulasi@gmail.com

Abstract - So many number of algorithms have been proposed for the discovery of data from the large database. However, since the number of generated patterns can be large, selecting which patterns to analyze can be nontrivial. There is thus a need for algorithms and tools that can assist in the selection of discovered patterns so that subsequent analysis can be performed in an efficient and, ideally, interactive manner. In this project, we propose a mark-based indexing method to optimize the storage and retrieval of a relative data's from the large database.

Keywords *- Content-based data mining queries, organizing temporal patterns, mark-based indexing methods.*

I. INTRODUCTION

Many rule discovery algorithms in data mining generate a large number of patterns/rules, sometimes even exceeding the size of the underlying database, with only a small fraction being of interest to the user [1]. However, when there are a large number of generated rules, identifying and analyzing those that are interesting becomes difficult.

In the case of association rule mining, several Approaches for the post processing of discovered association rules have been discussed. One approach is to group "similar" rules [3], which works well for a moderate number of rules only. Several query languages Mine-Rule [4], DMQL [5], and OLE DB [6] are designed for this purpose.

While most previous studies are focused on the post processing of association rules, this paper deals with the post processing of temporal patterns [7]. However, as the database grows, sequential scanning procedure can be too slow, and indexes should be built to speed up the queries. The problem is to determine what types of indexes are suitable for improving the speed of queries involving the content of temporal patterns.

This paper focuses on supporting content-based queries of temporal patterns, as opposed to point- or range-based queries. One example is the subpattern query: Given a set of patterns D and a query pattern q, find the temporal patterns in D that contain q. For example, we may wish to study further the behavior of a group of rules for which a particular pattern q is already known to be a component. To address this form of query, a mark-based indexing method is proposed that can speed up content-based queries on temporal patterns.

II. RELATED WORK

The temporal patterns described in this paper consist of two components: a set of states and a set of relationships between those states that represent the order of states within the pattern. In order to retrieve such patterns efficiently, any indexing method should deal with both temporal concepts—states and state relationships. Two signature file organizations are considered for set based attributes .They are sequential signature file and the bit-slice file.

III. PRELIMINARIES

A) PROBLEM DESCRIPTION

Definition 1 (state sequence) : Let S denotes the set of all possible states. A state $s \varepsilon S$ that holds during a period of time [b, f) is denoted as (b, s, f), where b is the start time, and f is the end time. The (b, s, f) triple is termed a state interval.

Definition 2(temporal pattern) : Given n state intervals (b_i,s_i,f_i), $1\leq i\leq n$, a temporal pattern of size n is defined by a pair (s,M), where $s:\{1,.., n\}\rightarrow S$ maps index i to the corresponding state, and M is an $n\times n$ matrix whose elements M[i,j] denote the relationship between intervals $[b_i,f_i)$ and $[b_j,f_j)$ The size of a temporal pattern α is the number of intervals in the pattern, denoted as dim(α) If the size of α is n, then α is called an n-pattern.

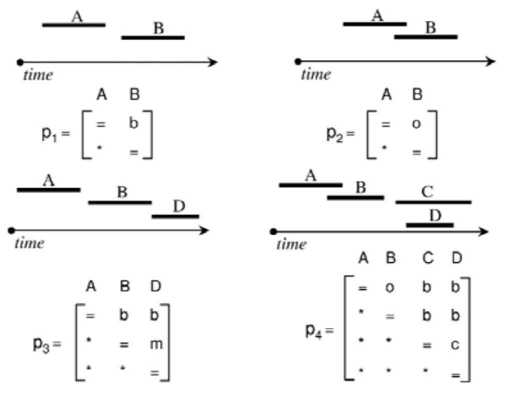
Fig. 1 shows four normalized temporal patterns defined over a set of states S= {A, B, C, D} and a set of interval relations Rel.

Definition 3 (sub pattern) : A temporal pattern $\alpha=(s_\alpha,M_\alpha)$ is a sub pattern of (s_β,M_β), denoted $\alpha\sqsubseteq\beta$, if $\dim(s_\alpha,M_\alpha)\leq\dim(s_\beta,M_\beta)$ and there is an injective mapping

$\pi:\{1,..,\dim(s_\alpha,M_\alpha)\} \rightarrow: \{1,..,\dim(s_\beta,M_\beta)\}$ such that $\forall$ i, j ε {1,.., dim (s_α,M_α)}:

$s_\alpha(i)= s_\beta(\pi(i)) \land M_\alpha[i,j]= M_\beta[\pi(i),\pi(j)]$.

Definition 4 (content-based queries) : Let D be a temporal pattern database and q be a query pattern. The four forms of content-based queries



$$p_1 = \begin{array}{c|cc} & A & B \\ \hline & = & b \\ & * & = \end{array} \quad p_2 = \begin{array}{c|cc} & A & B \\ \hline & = & o \\ & * & = \end{array}$$

$$p_3 = \begin{array}{c|ccc} & A & B & D \\ \hline & = & b & b \\ & * & = & m \\ & * & * & = \end{array} \quad p_4 = \begin{array}{c|cccc} & A & B & C & D \\ \hline & = & o & b & b \\ & * & = & b & b \\ & * & * & = & c \\ & * & * & * & = \end{array}$$

Fig.1 : Example of temporal patterns.

that this research supports include the following:

1. Subpattern queries.
2. Superpattern queries.
3. Equality queries
4. K-nearest subpattern queries

Superpattern queries are useful when searching for the characteristic parts of a large pattern, while k-nearest subpattern queries limit the number of patterns generated by subpattern or superpattern queries.

B) TEMPORAL PATTERN SIMILARITY

To answer k-nearest subpattern queries, a suitable measure of similarity among temporal patterns needs to be defined. Jaccard coefficient expresses the fraction of elements common to both sets.

Given two temporal patterns p_2 and p_3, let S_α and S_β denote a set of states in α and β, respectively. The similarity between patterns α and β is defined as follows:

$$\text{sim}(\alpha, \beta) =|S_c|+|R_c|/\sqrt{}\ (N_\alpha^s+ N_\alpha^r) \times (N_\beta^s+ N_\alpha^r).$$

Here, $|S_c| =|S_c \cap S_\beta|$ is the number of common states in α and β, and $|R_c|$ represents the number of common relationships. N_α^s represents the number of states (size) of α, and N_α^r represents the number of relationships in α. For a temporal pattern α of size n, N_α^s =n, and N_α^r=n(n-1)/2.The value of sim (α,β) will be 1 if α=β and will be 0 if they do not have common states. Therefore, the similarity between patterns p_2 and p_4 can be computed as

$$\text{sim}(P_2, P4) =2+1/\sqrt{}\ (2+1) \times (4+6) \approx 0.548.$$

By using this, the similarity matrix of the four temporal patterns in Fig. 1 is

		p1	p2	p3	p4
	P1	1	0.667	0.707	0.365
SIM=	P2	0.667	1	0.471	0.548
	P3	0.707	0.471	1	0.516
	P4	0.365	0.548	0.516	1

IV. SIGNATURE-BASED INDEX FOR RETRIEVAL OF TEMPORAL PATTERNS

This section describes the method used to construct the signature-based index from a collection of temporal patterns.

A) SIGNATURE FILES

A signature is a bit pattern formed for a data object, which is then stored in the signature file. Signature files were originally proposed in the context of word retrieval in text databases.

Three commonly used set-valued queries are

1. Subset query $(T \supseteq Q)$.
2. Superset query $(T \subseteq Q)$.
3. Equality query $(T \equiv Q)$.

An initial target signature is generated for each target set as follows: Each element in a target set is hashed to a bit pattern termed an element signature. All element signatures are of bit length F, and exactly b bits are set to “1,” where b < F. F is termed the length of a signature, while b is termed the weight of an element’s signature. A target signature is obtained by the bitwise union of all element signatures.

B) CONSTRUCTING SIGNATURE FILES FOR TEMPORAL PATTERNS

Two functions are required to create the equivalent set of a temporal pattern. The first function is used to map a set of states in the pattern into a set of integers, while the second maps the relationships between states into integers. These two functions can be defined as:

Definition 5 (state mapping) : Given a set of states S, a state mapping function f(x) is a function that transforms a state type x Є S into an integer value, such that $f(x) \neq f(y)$, where x, y Є S.

Definition 6 (relationship mapping) : Given a set of states S and a set of relations Rel, a relationship mapping g(x, y, r) is a function that transforms a relationship (x **r** y) into an integer value, where x, y Є S, and **r** Є Rel.

TABLE 1

Equivalent Sets and Signatures of Temporal Patterns

Pattern	Equivalent Set	Signature
P1	{1, 2, 30}	0100 0110
P2	{1, 2, 22}	0100 0110
P3	{1, 2, 4, 30, 32, 52}	0101 0111
P4	{1, 2, 3, 4, 22, 31, 32, 59, 60, 28}	1101 1111

Definition 7 (equivalent set) : Given a temporal pattern p of size k, S_p=<s1,…,s_2> is the list of states in p, and M_p is a k×k matrix whose element M_p[i,j] denotes the relationship between states s_i and s_j in Sp. The equivalent set of p,

E (p), is defined as

$$E(p)=\left(\bigcup_{i=1}^{k}\{f(s_i)\}\right)\cup\left(\bigcup_{i=1}^{k-1}\bigcup_{j=i+1}^{k}\{g(s_i,s_j,r)\}\right),$$

Where r=Mp[i,j]

For example, using the mapping functions f and g in the previous example, the equivalent set of patterns p1 and p3 (Fig. 1) can be computed as follows:

E(p1)={(f(A)} ∪{f(B)} ∪ {g(A,B,b)}={1,2,30},

E(p3)={(f(A)} ∪ {f(B)} ∪{f(D)} ∪{g(A,B,b)} ∪{g(A,D,b)} ∪ {g(B,D,m)}= {1,2,4,30,32,52}.

Equivalent sets of the other temporal patterns are shown in the second column of Table 1.

p1 is a sub pattern of p3; therefore, E (p1) ⊆E (p3) (Table 1). This property is formalized in the following.

Property 1 : Given two temporal patterns p and q and the corresponding equivalent sets E (p) and E (q), the following properties hold for any two temporal patterns and their equivalent sets:

1. p ⊒ q →E (p) ⊇E (q).

2. p ⊑ q →E (p) ⊆ E (q).

3. p =q →E (p) = E (q).

Definition 8 (signature) : The signature of an equivalent set E, denoted sig (E), is an F-bit binary number created by the bitwise union of all element signatures in E. For example, given F =8 and m=1, the signature of element e Є E is an 8-bit binary number that can be computed by a hash function hash (e) $=2^{(emodF)}$. For the set E(p3)={1,2,4,30,32,52},its element signatures are hash (1) =00000010; hash (2) =00000100;

hash (4) =00010000; hash (30) =01000000;

hash (32) =00000001; & hash (52) =00010000.

The signature of E (p_3) is computed using the bitwise union of all these element signatures, and the resulting signature is “01010111.”

Property 2: Given two equivalent sets E (p) and E (q) and their corresponding signatures sig_p and sig_q, the signatures of equivalent sets have the following properties:

1. E (p) ⊇E (q) → $sig_p \Lambda sig_q = sig_q$.
2. E (p) ⊆E (q) → $sig_p \Lambda sig_q = sig_q$.
3. E (p) =E (q) → $sig_p = sig_q$.

Using these methods, the signature file of temporal patterns in database D can be created as follows: For each temporal pattern p Є D, its equivalent set E (p) is calculated, and then, its signature E_p is generated. This signature, together with the temporal pattern identifier (pid), is inserted into the signature file. This procedure is outlined in Algorithm 4.1.

Algorithm 4.1: Constructing a signature file of temporal patterns

Input: A database D of temporal patterns

Output: SignatureFile

```
1:  for each p Є D do
2:      E (p) =Equivalent_Set (p)
3:      $sig_p$=Signature (E (p))
4:      Insert <$sig_p$, $pid_p$> into SignatureFile
5:  end for
6:  return SignatureFile
```

C) ANSWERING CONTENT-BASED QUERIES USING THE SIGNATURE FILE

i) SUB PATTERN QUERIES

Given a temporal pattern database D and a query pattern q, the algorithm for evaluating subpattern queries is called evaluateSubPattern (D, q).

ii) SUPERPATTERN QUERIES

Let evaluateSuperPattern (D, q) be the algorithm for evaluating super pattern queries, that is, for finding temporal patterns in D that are contained in q.

IV) K-NEAREST SUBPATTERN QUERIES

K-nearest queries are used to limit the number of patterns generated by sub pattern queries. Given that a sub pattern query generates n temporal patterns containing the query pattern q.

V) EXPERIMENTS

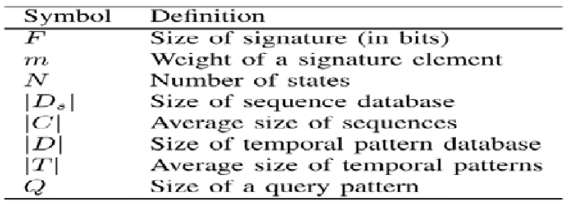
To assess the performance of the proposed methods, the evaluateSubPattern, evaluateSuperPattern, and evaluateEquality were implemented on SSF and BSSF signature files.

Experimental parameters are listed in Table 2.

All programs are written in Java Language. The experiments were conducted on synthetic data sets on a 1.3-GHz Intel Celeron PC with 384 Mbytes of RAM running Windows XP Professional.

TABLE 2

Parameters



Symbol	Definition
F	Size of signature (in bits)
m	Weight of a signature element
N	Number of states
$\|D_s\|$	Size of sequence database
$\|C\|$	Average size of sequences
$\|D\|$	Size of temporal pattern database
$\|T\|$	Average size of temporal patterns
Q	Size of a query pattern

A) EFFECT OF SIGNATURE SIZE ON THE NUMBER OF FALSE DROPS

The false-drop probability depends on F, m, and the cardinalities of the query set and target set. The size of the database |D|=50,000, the average size of temporal pattern |T|=5, and the number of states N=100.

The number of false drops is similar for both SSF and BSSF. Conversely, the larger the query pattern, the higher the number of false drops for evaluateSuperPattern (Fig. 2b).The best recorded performance improvement was achieved with a signature size between 16 and 32 bits, at which point the number of false drops decreases significantly.

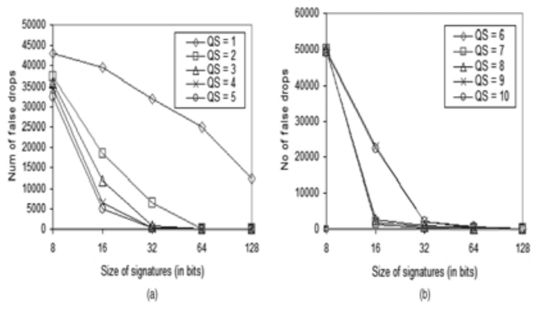


Fig. 2**:** Effect of signature size on the number of false drops.

(a) evaluateSubPattern,

(b) evaluateSuperPattern.

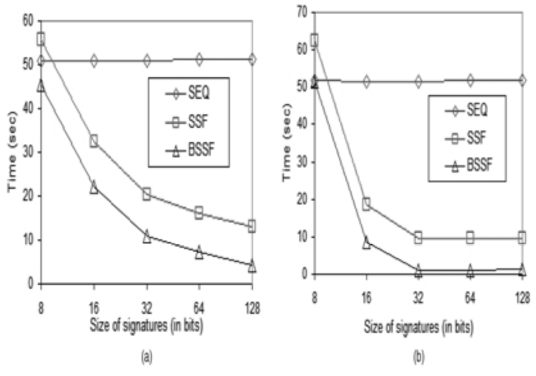


Fig.3. Effect of signature size on query processing time. (A) evaluateSubPattern. (b) evaluateSuperPattern.

B) EFFECT OF SIGNATURE SIZE ON QUERY PROCESSING TIME

Fig. 3a shows the total time required by evaluateSubPattern, while Fig. 3b shows the total time required by evaluateSuperPattern. Both methods gain the best performance improvement when the size of signature is between 16 and 32. Both methods perform better on BSSF. Finally, both methods show a marked improvement on SSF and BSSF over SEQ.

C) EFFECT OF DATABASE SIZE ON THE QUERY PROCESSING TIME

In order to observe how the methods scale with respect to the database size, five data sets were generated in which |T|=5 and N=100. The size of database |D| was varied from 10,000 to 100,000.

Fig. 4a shows the total time required by evaluateSubPattern on SEQ, SSF 32 bits, SSF 64 bits, BSSF 32 bits, and BSSF 64 bits to process five queries.

Fig. 4b shows the total time required by evaluateSuperPattern to process the five queries.

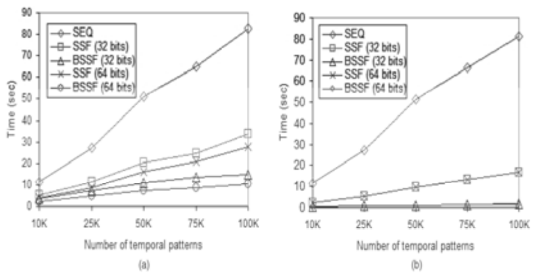


Fig. 4 : Effect of database size on query processing time. (a) evaluate Sub Pattern. (b) evaluate Super Pattern.

D) EXPERIMENTS ON REAL DATA

A temporal pattern database was generated by running ARMADA on this interval sequence database and setting the minimum support to 1 percent and the minimum gap to 100. The resulting temporal pattern database contains 210,580 frequent temporal patterns, as summarized in Table 3.

TABLE 3

Generated Temporal Patterns from the ASL Database

Pattern size	Number of patterns
1	177
2	8768
3	65178
4	100847
5	32221
6	3239
7	150
Total	210580

VI. CONCLUSION AND FUTURE WORK

The use of a signature-based index for content-based retrieval of temporal patterns has been presented. The signatures of temporal patterns are created by first converting temporal patterns into equivalent sets and then generating the signatures from the equivalent sets.

In conclusion, the use of signature files improves the performance of temporal pattern retrieval. The bit-slice signature file performs better than the SSF and is a good choice for content-based retrieval of temporal patterns. This retrieval system is currently being combined with visualization techniques for monitoring the behavior of a single pattern or a group of patterns over time.

REFERENCES

[1] T. Imielinski and A. Virmani, “Association Rules . . . and what’s next? Towards Second Generation Data Mining Systems,” Proc.

[2] L. Geng and H.J. Hamilton, “Interestingness Measures for Data Mining: A Survey,” ACM Computing Surveys, vol. 38, no. 3, 2006.

[3] H. Toivonen, M. Klemettinen, P. Ronkainen, K. Hatonen, and H. Mannila, “Pruning and Grouping of Discovered Association Rules,” Proc. ECML Workshop Statistics, Machine Learning, and Knowledge Discovery in Databases, pp. 47-52, 1995.

[4] R. Meo, G. Psaila, and S. Ceri, “A New SQL-Like Operator for Mining Association Rules,” Proc. 22nd Int’l Conf.

[5] J. Han, J.Y. Chiang, S. Chee, J. Chen, Q. Chen, S. Cheng, W. Gong, M. Kamber, K. Koperski, G. Liu, Y. Lu, N. Stefanovic.

[6] A. Netz, S. Chaudhuri, U.M. Fayyad, and J. Bernhardt, “Integrating Data Mining with SQL Databases: OLE DB for Data Mining,” Proc. 17th Int’l Conf. Data Eng. (ICDE’01), pp. 379-387, 2001.

[7] C.M. Antunes and A.L. Oliveira, “Temporal Data Mining: An Overview,” Proc. ACM SIGKDD Workshop Temporal Data Mining, pp. 1-13, 2001.

[8] S. Helmer and G. Moerkotte, “A Performance Study of Four Index Structures for Set-Valued Attributes of Low Cardinality,” VLDB J., vol. 12, no. 3, pp. 244-261, 2003.

[9] T. Morzy and M. Zakrzewicz, “Group Bitmap Index: A Structure for Association Rules Retrieval,” Proc. ACM SIGKDD ’98.