

# $TSET^{MAX}$ : An Algorithm for Mining Frequent Maximal Temporal Patterns

Francisco Guil, Alfonso Bosch  
Dept. Lenguajes y Computación  
Universidad de Almería  
{fguil, abosch}@ual.es

Roque Marín  
Dept. Ing. de la Inf. y las Comunicaciones  
Universidad de Murcia  
roque@dif.um.es

## Abstract

*The incorporation of temporal semantics into the traditional data mining techniques has caused the creation of a new area called Temporal Data Mining. This incorporation is especially necessary if we want to extract useful knowledge from dynamic domains, which are time-varying in nature. However, in a lot of cases is practically a computationally intractable problem and therefore it poses more challenges on efficient processing than non-temporal techniques. Based in the inter-transactional framework, in [13] we proposed an algorithm named  $TSET$  for mining temporal patterns (sequences) from datasets. One of the main drawbacks of this algorithm is the fact that it is an Apriori-style and therefore, for each frequent pattern, all subsets of it need to be generated. In datasets with long patterns, this could degenerate into a computationally unfeasible problem. To address this problem, in this paper we present a new algorithm named  $TSET^{MAX}$  for mining frequent maximal temporal patterns. This algorithm extract the patterns using a lookahead technique and a depth first search on a temporal set-enumeration tree.*

## 1. Introduction

Data mining is an essential step in the process of knowledge discovery in databases that consists of applying data analysis and discovery algorithms that produce a particular enumeration of structures over the data [11]. There are two types of structures: models and patterns. So, we can talk about local and global methods in data mining [20]. In the case of local methods, the simplest case of pattern discovery is finding *association rules* [4]. The initial motivation for association rules was to aid in the analysis of large transactional databases. The discovery of association rules can potentially aid decision making within organizations. Another approach is integrating the data mining process into the development of Knowledge Based Systems [22].

Since the problem of mining association rules was in-

troduced by *Agrawal* in [4], a large amount of work has been done in several directions, including improvement of the *Apriori* algorithm, mining generalized, multi-level, or quantitative association rules, mining weighted association rules, fuzzy association rules mining, constraint-based rule mining, efficient long patterns mining, maintenance of the discovered association rules, etc. We want to point out the work in which a new type of association rules was introduced, the *inter-transaction association rules* [19, 18]. Temporal data mining can be viewed as an extension of this work.

Temporal data mining can be defined as the activity of looking for interesting correlations or patterns in large sets of temporal data accumulated for other purposes [9]. It has the capability of mining activity, inferring associations of contextual and temporal proximity, some of which may also indicate a cause-effect association. This important kind of knowledge can be overlooked when the temporal component is ignored or treated as a simple numeric attribute [25].

Data mining is an interdisciplinary area which has received contributions from a lot of disciplines, mainly from databases, machine learning and statistic. In [29] we found a review of three books, each one written from a different perspective. Although each perspective make strong emphasis on different aspects of data mining (efficiency, effectiveness, and validity), only when we simultaneously take these three aspects into account we may get successful data mining results. However, in the case of temporal data mining techniques, the most influential area is artificial intelligence because its work in temporal reasoning have guided the development of many of this techniques. In non-temporal data mining techniques, there are usually two different tasks, the description of the characteristics of the database (or analysis of the data) and the prediction of the evolution of the population. However, in temporal data mining this distinction is less appropriate, because the evolution of the population is already incorporated in the temporal properties of the data being analyzed.

We can found in the literature a large quantity of temporal data mining techniques. We want to highlight some

of the most representative ones. So, we can talk about sequential pattern mining [5], episodes in event sequences [21], temporal association rules mining [6, 14, 15], discovering calendar-based temporal association rules [16], patterns with multiple granularities mining [9], and cyclic association rules mining [23]. However, there is an important form of temporal associations which are useful but could not be discovered with this techniques. These are the inter-transaction associations presented in [18, 19]. The introduction of this type of associations was motivated by the observation that many real-world associations happen under certain context, such as time, place, etc. In the case of temporal context, inter-transactional associations represents associations amongst items along the dimension of time. Due to the number of potential association becomes extremely large, the mining of inter-transaction association poses more challenges on efficient processing than classical approaches. In order to make the mining of inter-transaction associations practical and computationally tractable, several methods have been proposed in [27, 26, 12].

Working in the same direction, in [13] we presented an algorithm named *TSET* based on the inter-transactional framework for mining frequent sequences (also called frequent temporal patterns or frequent temporal associations) from several kind of datasets. The improvement of the proposed solution was the use of a unique structure to store all frequent sequences. The data structure used is the well-known set-enumeration tree, commonly used in the data mining area [7, 3, 8, 10], in which the temporal semantic is incorporated.

*TSET* follows the same basic principles as most apriori-based algorithms [4]. Frequent sequence mining is an iterative process, and the focus is on a *level-wise* pattern generation. This implies that for extract a frequent sequence of length  $k$ , it must produce all  $2^k - 2$  of its subsequences since they must be frequent too (downward closure property). This exponential complexity causes the inadequacy of the method in datasets with long associations. Some of the algorithms in the literature avoid this implementing lookahead techniques, in which supersets of frequent patterns are used in order to prune off potential candidates. As we can see in [2], there are two type of methods, lattice-based methods, such as MaxEclat or Pincer-Search [28, 17], and tree-based methods, such as MaxMiner or DepthProject [7, 1]. However, all this algorithm are applicable to datasets belonging to non-temporal domains.

The aim of this paper is to propose a new algorithm named *TSET<sup>MAX</sup>* for mining frequent maximal sequences from several kind of datasets. Just like *TSET*, it uses a unique tree-based structure, the temporal set-enumeration tree. However, in this case, a depth first search on the tree with a lookahead technique is used. We want to point out that from the frequent maximal patterns dis-

covered we can generate its subsets and count their support by reading the database once. It is very important if the goal of the mining process is, for example, to extract all association rules.

The remainder of this paper is organized as follows. Section 2 gives a formal description of the problem of mining frequent maximal temporal patterns (maximal sequences) from datasets. Section 3 introduces the algorithm named *TSET<sup>MAX</sup>*. Section 4 presents an example to illustrate the basic ideas of the proposed algorithm. Conclusions and future works are finally drawn in Section 5.

## 2. Problem Description

In this section we describe our notation, some basic definition, and we introduce the goals of our algorithm.

**Definition 1** A dataset  $D$  is an ordered sequence of records  $D[0], D[1], \dots$ , where each  $D[i]$  can have  $col$  attributes,  $c[0], \dots, c[col-1]$ . The 0-attribute will be the dimensional attribute, the temporal data associated with the record, expressed in temporal units. The rest of attributes can be quantitative or categorical. We assume that the domain of each attribute is a finite subset of non-negative integers, and we also assume that the structure of time is discrete and linear. Due to every event registered has its absolute date identified, we represent the time for events with an absolute dating system [24]. In order to simplify the calculations, we transform the original dataset subtracting the date of each record from the date of the first record, the time origin.

With this generic definition of dataset, and with minimal modifications, the algorithm that we propose works with different types of sources, that is, relational databases, transactional databases and data streams.

**Definition 2** An event  $e$  is a 3-tuple  $(c[i], v, t)$ , where  $0 < i < col$ ,  $v \in dom\{c[i]\}$ , and  $t \in dom\{c[0]\}$ , that is,  $t \in \mathbb{N}$ . Events are "things that happen", and they usually represent the dynamic aspect of the world [24]. In our case, an event is related to the fact that a value  $v$  is assigned to a certain attribute  $c[i]$  with the occurrence time  $t$ . We will use the notation  $e.c$ ,  $e.v$ , and  $e.t$  to set and get the attribute, value, and time variables related to the event  $e$ . The set of all distinct pairs  $(c[i], v)$  can be also called event types.

**Definition 3** Given two events  $e_1$  and  $e_2$ , we define the  $\leq$  relation as follows:

1.  $e_1 = e_2$  iff  $(e_1.t = e_2.t) \wedge (e_1.c = e_2.c) \wedge (e_1.v = e_2.v)$
2.  $e_1 < e_2$  iff  $(e_1.t < e_2.t) \vee ((e_1.t = e_2.t) \wedge (e_1.c < e_2.c))$

We assume that a lexicographic ordering exists among the pairs (attribute, value) in the dataset.

**Definition 4** A sequence (or event sequence) is an ordered set of events  $S = \{e_0, e_1, \dots, e_{k-1}\}$ , where for all  $i < j$ ,  $e_i < e_j$ . Obviously,  $|S| = k$ . Note that different events with the same temporal unit can belong to the same sequence. Also, the same events with different temporal unit associated can belong to the same sequence. But nevertheless, in any sequence there will exist two or more pairs (attribute, value) associated to the same temporal unit. In other words, an attribute can not take two different values in the same instant.

**Definition 5** Let  $U_{tmin}$  be the minimal dimensional value associated to the sequence  $S$ . In other words,  $U_{tmin} = \min\{e_i.t\}$ , for  $e_i \in S$ . If  $U_{tmin} = 0$ , we say that  $S$  is a normalized sequence. Note that any non-normalized sequence can be transformed into a normalized one through a normalization function.

**Example 1** Let  $S_1 = \{(0, 0, 0), (1, 0, 0), (3, 0, 2)\}$ , and  $S_2 = \{(0, 0, 3), (1, 0, 3), (3, 0, 5)\}$  be two sequences.  $S_1$  is a normalized sequenced, since it has the minimal value equal to 0 for the temporal dimension. But  $S_2$  is not a normalized sequence, because its minimal value is not equal to zero. However, we can normalize  $S_2$  by subtracting its minimal value ( $U_{tmin} = 3$ ) from the temporal values as follows:  $S'_2 = \{(0, 0, 3 - 3), (1, 0, 3 - 3), (3, 0, 5 - 3)\}$ , resulting in the normalized sequence  $S'_2 = \{(0, 0, 0), (1, 0, 0), (3, 0, 2)\}$ .

Let  $U_{tmax}$  be the maximal dimensional value associated to the sequence  $S$ . This value indicates the maximum distance amongst the events belonging to the normalized sequence  $S$ . In other words,  $U_{tmax} = e_k.t$ , where  $|S| = k$ . From both, confidence and complexity points of view [18], this value will be always less or equal than a user-defined parameter called *maxspan*, denoted by  $w$ .

**Definition 6** The support (frequency) of a sequence is defined as:  $support(S) = \frac{|D_S|}{|D|}$ , where  $|D_S|$  denotes the number of occurrences of the sequence  $S$  in the dataset, and  $|D|$  is the number of records in the dataset  $D$ .

**Definition 7** A frequent sequence is a normalized sequence whose support is greater or equal than a user-specified threshold called minimum support *minsup*.

**Definition 8** A sequence is a frequent maximal sequence if and only if it is frequent and no proper super-sequence (superset) of it is frequent.

Given a dataset and *minsup*, the goal of maximal temporal pattern (or sequence) mining is to determine in the dataset all the frequent maximal sequences whose support are greater than or equal to *minsup*.

### 3 The $TSET^{MAX}$ Algorithm

In this section we describe the algorithm that we propose for the mining of frequent maximal sequences from a dataset.

```

algorithm  $TSET^{MAX}$ (dataset  $D$ , minsup  $m$ , maxspan  $w$ )
begin
  tree.init( $D$ ,  $m$ ,  $w$ );
  tree.getSequences( $D$ ,  $m$ ,  $w$ );
  Output(tree);
end;

```

Figure 1.  $TSET^{MAX}$  Algorithm

The first step consists in the creation and initialization of the data structure. The "init" method that outlines Figure 2 returns the frequent 1-sequences (frequent events) found in the dataset. It uses a Binary Search Tree auxiliary structure to compute effectively the support of the events presented in the dataset. The inorder traversal of this structure produces the 1-sequences lexicographically ordered. The "getSequences" method (see Figure 3) obtain iteratively the frequent k-sequences using a queue structure. After generating the candidate set, which consists in copying all the events to the right of a given event in the newNode, and pruning the infrequent sequences, the new node is pushing into the queue for the next kth-cycle.

```

procedure init(tree, dataset  $D$ , minsup  $m$ , maxspan  $w$ )
begin
  BST auxiliarTree =  $\emptyset$ ;
  foreach record  $i$  in  $D$ 
    foreach non-dimensional attribute  $j$  in record
      auxiliarTree.insert( $D[i].c[j]$ ,  $D[i].c[j].v$ ,  $D[i].c[0]$ );
  tree.insert(auxiliarTree.traverseInOrder( $m$ ));
end;

```

Figure 2. The code for the method init

We will use an example to illustrate the data structure employed by  $TSET^{MAX}$ . But, before doing it, let us introduce new definitions.

**Definition 9** An Extended Set-Enumeration Tree is an set of nodes, where each node can be:

- The empty set.
- A root node of  $n$  disjoint trees.

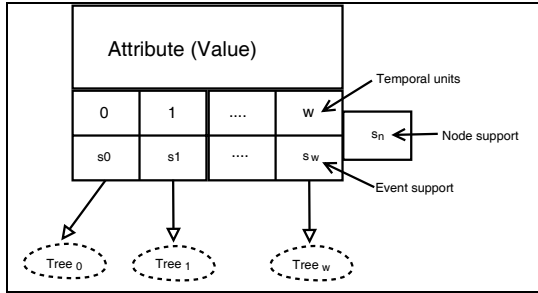
```

procedure getSequences(tree, dataset  $D$ , minsup  $m$ , maxspan  $w$ )
begin
  Queue  $Q = \emptyset$ ;
   $Q$ .push(tree.root);
  while ( $Q \neq \emptyset$ )
    begin
      act =  $Q$ .pop();
      foreach node  $n$  in act
        if  $n$  is maximal then continue;
        newNode =  $n$ .getCandidates();
        newNode.evaluateSupport( $D$ ,  $w$ );
        newNode.pruningInFrequent( $m$ );
        if(newNode  $\neq \emptyset$ )
          begin
             $n$ .child = newNode;
             $Q$ .push(newNode);
          end;
        end;
      end;
    end;
end;

```

**Figure 3. The code for the method getSequences**

**Definition 10** A node is an set of events, that is,  $node = (e_0, e_1, \dots, e_{n-1})$ , where  $e_i = (c[j], v, t, s)$ ,  $c[j]$  the attribute,  $v$  the value assigned to this attribute,  $t$  the occurrence time of the event, and  $s$  the support value associated. In Figure 4 we can see an schematic representation of a node.



**Figure 4. An schematic representation of a node**

**Definition 11** Let  $N = (e_0, e_1, \dots, e_{i-1})$  be a node of the tree. Each event  $e_i$  will have a null link or a link to a subtree rooted by the node  $N_{e_i} = (e_0, e_1, \dots, e_{j-1})$ . We will say that  $e_i$  is the parent of  $N_{e_i}$ , or simply,  $e_i = P(N_{e_i})$ . Conversely,  $N_{e_i}$  is the child of  $e_i$ , or  $N_{e_i} = C(e_i)$ . By definition, for  $e_j \in N_{e_i}$ ,  $P(e_j) = P(N_{e_i}) = N$ .

**Table 1. An example dataset**

T	$C_0$	$C_1$
0	0	0
1	1	1
2	0	0
3	1	1

This definition of ancestral relationship naturally defines the tree structure of the nodes, which is rooted at the *null* node. Our goal is to use this structure in a effective way in order to reduce the nodes generated and, consequently, the CPU time for extract frequent maximal sequences.

**Definition 12** Let  $N = (e_0, e_1, \dots, e_{j-1})$  be a node. The Head of the node  $N$  is defined as follows:  $H(N) = (e_{i_0}, e_{i_1}, \dots, e_{i_{k-1}})$ , where  $\forall e_{i_j}, 0 \leq j < k - 1, e_{i_j} = P(e_{i_{j+1}})$ . By definition, for each  $e_j \in N, H(e_j) = H(N)$ . We denote the length of  $H(N)$  as  $l$ , that is,  $l = \text{length}(H(N))$

We want to point out that for each  $e_j \in N, H(e_j) \cup e_j$  corresponds with a branch of the tree of length  $l+1$ , in other words, corresponds with a sequence of length  $l+1$ . The support value associated with the event  $e_j$  is the support of the sequence  $H(e_j) \cup e_j$ .

**Definition 13** A node  $N = (e_0, e_1, \dots, e_{i-1})$  is a possibly maximal node if and only if for each  $e_j, 0 \leq h < i - 1, e_j < e_{j+1}$  (total order).

**Definition 14** A node  $N = (e_0, e_1, \dots, e_{i-1})$  is a maximal node if  $N$  is possibly maximal and the support of  $H(N) \cup N$  is greater than the user-defined minsup.

$H(N) \cup N$  is a maximal sequence if and only if  $N$  is a maximal node and no proper superset of it is frequent.

The last definitions support the basic idea of the algorithm. The algorithm combines both subset infrequency and superset frequency properties pruning strategies to remove branches from consideration [7]. If  $N$  is a maximal node, the algorithm does not decompose it into subnodes, because  $H(N) \cup N$  is a superset of all possible nodes generated from it. The deep first strategy quickly tends to find the maximal patterns first in the search process [1].

## 4. Example

We will use a synthetic dataset to illustrate the basic ideas of the algorithm that we propose in this work. Suppose that, after some preprocessing, we have obtained the dataset  $D$  shown in Table 1, where  $T$  represents the temporal attribute, and  $C_0$  and  $C_1$  two descriptive attributes. We assume that *minsup* is set at 50% and *maxspan* at 1 temporal unit.

First, we show the nodes studied by the *TSET* algorithm. As we said before, *TSET* is an *Apriory* – style algorithm and, therefore, it uses a breadth-first and level-wise strategy for extract the frequent sequences. For each node, we will show only the frequent events associated.

Let  $N_0$  be the initial node, that is, the set of frequent 1-sequences.

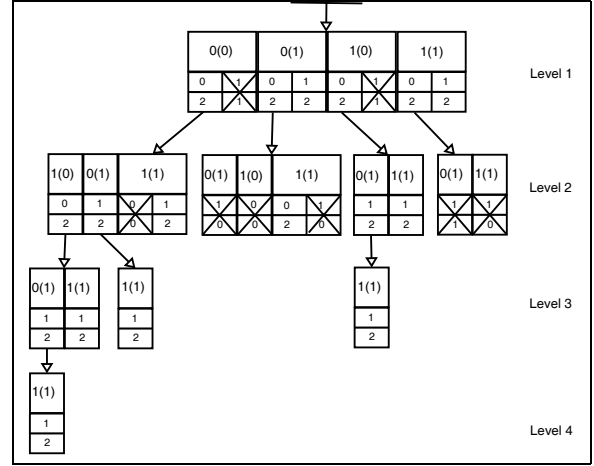
- $H(N_0) = null$   
 $N_0 = ((C_0, 0, 0, 2), (C_0, 1, 0, 2), (C_1, 0, 0, 2), (C_1, 1, 0, 2), (C_0, 1, 1, 2), (C_1, 1, 1, 2))$
- $H(N_1) = ((C_0, 0, 0, 2))$   
 $N_1 = ((C_1, 0, 0, 2), (C_0, 1, 1, 2), (C_1, 1, 1, 2))$
- $H(N_2) = ((C_0, 1, 0, 2))$   
 $N_2 = ((C_1, 1, 0, 2))$
- $H(N_3) = ((C_1, 0, 0, 2))$   
 $N_3 = ((C_0, 1, 1, 2), (C_1, 1, 1, 2))$
- $H(N_4) = ((C_1, 1, 0, 2))$   
 $N_4 = null$
- $H(N_5) = ((C_0, 0, 0, 2), (C_1, 0, 0, 2))$   
 $N_5 = ((C_0, 1, 1, 2), (C_1, 1, 1, 2))$
- $H(N_6) = ((C_0, 0, 0, 2), (C_0, 1, 1, 2))$   
 $N_6 = ((C_1, 1, 1, 2))$
- $H(N_7) = ((C_1, 0, 0, 2), (C_0, 1, 1, 2))$   
 $N_7 = ((C_1, 1, 1, 2))$
- $H(N_8) = ((C_0, 0, 0, 2), (C_1, 0, 0, 2), (C_0, 1, 1, 2))$   
 $N_8 = ((C_1, 1, 1, 2))$

Figure 5 shows the temporal set-enumeration tree generated by the *TSET* algorithm. This structure stores a total of 15 frequent sequences.

Now, we show how the *TSET*<sup>MAX</sup> algorithm works. As we said before, *TSET*<sup>MAX</sup> uses a deep-first search and a lookahead strategy for mining frequent maximal sequences.

- $H(N_0) = null$   
 $N_0 = ((C_0, 0, 0, 2), (C_0, 1, 0, 2), (C_1, 0, 0, 2), (C_1, 1, 0, 2), (C_0, 1, 1, 2), (C_1, 1, 1, 2))$

Since  $N_0$  is not a possible maximal node, the algorithm generates new candidate nodes, copying all the events to the right of a given event in the new nodes. As we want to extract normalized sequences, the 1-sequences  $(C_0, 1, 1, 2)$ , and  $(C_1, 1, 1, 2)$  will not generate new nodes.



**Figure 5. The extended set-enumeration tree structure generated by *TSET***

- $H(N_1) = ((C_0, 0, 0, 2))$   
 $N_1 = ((C_1, 0, 0, 2), (C_0, 1, 1, 2), (C_1, 1, 1, 2))$

Here, we want to highlight that the candidate node was:

$$N_{1_{old}} = ((C_1, 0, 0, 2), (C_1, 1, 0, 0), (C_0, 1, 1, 2), (C_1, 1, 1, 2))$$

which is not a possible maximal node (as we can see in the events one and two, has two different values are assigned to  $C_1$  in the same instant). However, after pruning, the node  $N_1$  becomes a possible maximal, and after the counting process, it support value is assigned to 2, that is, it is a maximal node. As  $H(N_1) \cup N_1$  does not has a frequent superset, we just found a frequent maximal sequence:

$$S_1 = ((C_0, 0, 0, 2), (C_1, 0, 0, 2), (C_0, 1, 1, 2), (C_1, 1, 1, 2)), s = 2$$

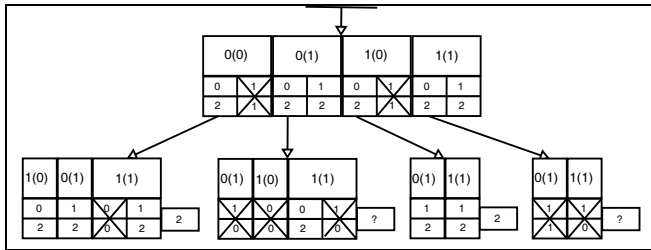
- $H(N_2) = ((C_0, 1, 0, 2)), s = 2$   
 $N_2 = ((C_1, 1, 0, 2))$

The frequent sequence  $H(N_2) \cup N_2$  has a frequent superset ( $S_1$ ). Therefore, it is discarded.

- $H(N_3) = ((C_1, 0, 0, 2))$   
 $N_3 = ((C_0, 1, 1, 2), (C_1, 1, 1, 2)), s = 2$

As in the previous case, this maximal node is discarded too.

Figure 6 shows the temporal set-enumeration tree generated by the *TSET*<sup>MAX</sup> algorithm. In this case, this structure stores a total of 3 maximal nodes, but only one frequent maximal sequence.



**Figure 6. The extended set-enumeration tree structure generated by  $TSET^{MAX}$**

## 5. Conclusions and Future Works

In this paper, we have presented  $TSET^{MAX}$ , an effective algorithm that uses a lookahead technique and deep first search on a tree-based data structure to discover frequent maximal sequences (maximal inter-transactional associations) from datasets. Although the enhancement of the method is significantly superior, in terms of computing effort, mining this type of associations is still a computationally intensive problem. Several optimizations techniques are being devised to speed up the discovery of sequences. In particular, we are working in the adaptation of the techniques proposed in the literature to reduce the number of database passes, and therefore develop an algorithm which can deal with large and dense synthetic and real datasets in an efficient way.

## References

- [1] R. C. Agarwal, C. C. Aggarwal, and V. V. V. Prasad. Depth first generation of long patterns. In *Proc. of the 6th ACM SIGKDD Int. Conf. on Knowledge discovery and Data Mining, August 20-23, Boston, MA, USA, 2000*, pages 108–118. ACM, 2000.
- [2] R. C. Agarwal, C. C. Aggarwal, and V. V. V. Prasad. A tree projection algorithm for generation of frequent item sets. *Parallel and Distributed Computing*, 61:350–371, 2001.
- [3] C. C. Aggarwal. Towards long pattern generation in dense databases. *SIGKDD Explorations*, 3(1):20–26, 2001.
- [4] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia, editors, *Proc. of the ACM SIGMOD Int. Conf. on Management of Data, Washington, D.C., May 26-28, 1993*, pages 207–216. ACM Press, 1993.
- [5] R. Agrawal and R. Srikant. Mining sequential patterns. In P. S. Yu and A. L. P. Chen, editors, *Proc. of the 11th Int. Conf. on Data Engineering, March 6-10, 1995, Taipei, Taiwan*, pages 3–14. IEEE Computer Society, 1995.
- [6] J. M. Ale and G. H. Rossi. An approach to discovering temporal association rules. In *Proc. of the 2000 ACM Symposium on Applied Computing, Villa Olmo, Via Cantoni 1, 22100 Como, Italy, March 19-21, 2000*, pages 294–300. ACM, 2000.
- [7] R. J. Bayardo. Efficiently mining long patterns from databases. In L. M. Haas and A. Tiwary, editors, *Proc. of the ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 1998), June 2-4, 1998, Seattle, Washington, USA*, pages 85–93. ACM Press, 1998.
- [8] F. Berzal, J. C. Cubero, N. Marín, and J. M. Serrano. TBAR: An efficient method for association rule mining in relational databases. *Data & Knowledge Engineering*, 37:47–64, 2001.
- [9] C. Bettini, X. S. Wang, and S. Jajodia. Testing complex temporal relationships involving multiple granularities and its application to data mining. In *Proc. of the 15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 3-5, 1996, Montreal, Canada*, pages 68–78. ACM Press, 1996.
- [10] F. Coenen, G. Goulbourne, and P. Leng. Tree structures for mining association rules. *Data Mining and Knowledge Discovery*, 8:25–51, 2004.
- [11] U. Fayyad, G. Piatetky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *AIMagazine*, 17(3):37–54, 1996.
- [12] L. Feng, J. X. Yu, H. Lu, and J. Han. A template model for multidimensional inter-transactional association rules. *The VLDB Journal*, 11:153–175, 2002.
- [13] F. Guil, A. Bosch, and R. Marín. TSET: An algorithm for mining frequent temporal patterns. In *Proc. of the First Int. Workshop on Knowledge Discovery in Data Streams, in conjunction with the ECML/PKDD 2004 Conference*, pages 65–74, 2004.
- [14] C. H. Lee, C. R. Lin, and M. S. Chen. On mining general temporal association rules in a publication database. In N. Cercone, T. Y. Lin, and X. Wu, editors, *Proc. of the 2001 IEEE Int. Conf. on Data Mining, 29 November - 2 December 2001, San Jose, California, USA*, pages 337–344. IEEE Computer Society, 2001.
- [15] J. W. Lee, Y. J. Lee, H. K. Kim, B. H. Hwang, and K. H. Ryu. Discovering temporal relation rules mining from interval data. In *Proc. of the 1st EurAsian Conf. on Information and Communication Technology (Eurasia-ICT 2002), Shiraz, Iran, October 29-31, 2002*, volume 2510 of *Lecture Notes in Computer Science*, pages 57–66. Springer, 2002.
- [16] Y. Li, P. Ning, X. S. Wang, and S. Jajodia. Discovering calendar-based temporal association rules. *Data & Knowledge Engineering*, 44:193–218, 2003.
- [17] D. Lin and Z. M. Kedem. Pincer-search: An efficient algorithm for discovering the maximum frequent set. *IEEE Transactions on Knowledge and Data Engineering*, 14(3):553–566, 2002.
- [18] H. Lu, L. Feng, and J. Han. Beyond intra-transaction association analysis: Mining multi-dimensional inter-transaction association rules. *ACM Transactions on Information Systems (TOIS)*, 18(4):423–454, 2000.
- [19] H. Lu, J. Han, and L. Feng. Stock movement and n-dimensional inter-transaction association rules. In *Proc. of the Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'98), Seattle, Washington, June 1998*, pages 12:1–12:7, 1998.
- [20] H. Mannila. Local and global methods in data mining: Basic techniques and open problems. In P. Widmayer, F. Triguero,

- R. Morales, M. Hennessey, S. Eidenbenz, and R. Conejo, editors, *In Proc. of the 29th Int. Colloquium on Automata, Languages and Programming (ICALP 2002), Malaga, Spain, July 8-13, 2002*, volume 2380 of *Lecture Notes in Computer Science*, pages 57–68. Springer, 2002.
- [21] H. Mannila, H. Toivonen, , and A. I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.
- [22] C. Ordonez, C. A. Santana, and L. de Braal. Discovering interesting association rules in medical data. In D. Gunopulos and R. Rastogi, editors, *Proc. of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, Dallas, Texas, USA, May 14, 2000*, pages 78–85, 2000.
- [23] B. Özden, S. Ramaswamy, and A. Silberschatz. Cyclic association rules. In *Proc. of the 14th Int. Conf. on Data Engineering, February 23-27, 1998, Orlando, Florida, USA*, pages 412–421. IEEE Computer Society, 1998.
- [24] A. K. Pani. Temporal representation and reasoning in artificial intelligence: A review. *Mathematical and Computer Modelling*, 34:55–80, 2001.
- [25] J. F. Roddick and M. Spiliopoulou. A survey of temporal knowledge discovery paradigms and methods. *IEEE Transactions on Knowledge and Data Engineering*, 14(4):750–767, 2002.
- [26] A. K. H. Tung, H. Lu, J. Han, and L. Feng. Breaking the barrier of transactions: Mining inter-transaction association rules. In *Proc. of the 5th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA*, pages 297–301. ACM Press, 1999.
- [27] A. K. H. Tung, H. Lu, J. Han, and L. Feng. Efficient mining of intertransaction association rules. *IEEE Transactions on Knowledge and Data Engineering*, 15(1):43–56, 2003.
- [28] M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New algorithms for fast discovery of association rules. In D. H. H. Mannila and D. Pregibon, editors, *Proc. of the 3rd Int. Conf. on Knowledge Discovery and Data Mining (KDD-97), Newport Beach, California, USA, August 14-17, 1997*, pages 283–286. AAAI Press, 1997.
- [29] Z. H. Zhou. Three perspectives of data mining (book review). *Artificial Intelligence*, 143:139–146, 2003.