Under consideration for publication in Knowledge and Information
Systems

Mining Frequent Arrangements of Temporal Intervals

Panagiotis Papapetrou[1], George Kollios[1], Stan Sclaroff[1] and Dimitrios Gunopulos[2]

[1]Department of Computer Science, Boston University, Boston MA, USA; [2]Department of Informatics and Telecommunications, University of Athens, Athens, Greece

Abstract. The problem of discovering frequent arrangements of temporal intervals is studied. It is assumed that the database consists of sequences of events, where an event occurs during a time-interval. The goal is to mine temporal arrangements of event intervals that appear frequently in the database. The motivation of this work is the observation that in practice most events are not instantaneous but occur over a period of time and different events may occur concurrently. Thus, there are many practical applications that require mining such temporal correlations between intervals including the linguistic analysis of annotated data from American Sign Language as well as network and biological data. Three efficient methods to find frequent arrangements of temporal intervals are described; the first two are tree-based and use breadth and depth first search to mine the set of frequent arrangements, whereas the third one is prefix-based. The above methods apply efficient pruning techniques that include a set of constraints that add user-controlled focus into the mining process. Moreover, based on the extracted patterns a standard method for mining association rules is employed that applies different interestingness measures to evaluate the significance of the discovered patterns and rules. The performance of the proposed algorithms is evaluated and compared with other approaches on real (American Sign Language annotations and network data) and large synthetic datasets.

Keywords: Sequential Pattern Mining; Data Mining; Temporal Mining; Arrangement Mining; Constraint-based Mining; Arrangement Rule Mining; American Sign Language.

1. Introduction

Sequential pattern mining has received particular attention in the last decade (Agrawal & Srikant, 1994; Agrawal & Srikant, 1995; Ayres, et al., 2002; Ba-

Received May 23 2008
Revised Oct 11 2008
Accepted Jan 17 2009

yardo, 1998; Zaki, 2001; Pei, et al., 2001; Pei, et al., 2002; Han, et al., 2000; Seno & Karypis, 2002; Yan, et al., 2003; Leleu, et al., 2003; Han, et al., 2000b; Wang & Han, 2004). The objective is to extract patterns from a set of sequences of instantaneous events which satisfy some user-specified constraints. These constraints can vary from just a support threshold, that defines frequency, to a set of gap, window (Zaki, 2000; Srikant & Agrawal, 1996), or regular expression constraints (Garofalakis, et al., 1999), that apply more user-controlled focus into the mining process. Despite advances in this area, nearly all proposed algorithms concentrate on the case where events occur at single time instants. However, in many applications events are not instantaneous; they instead occur over a time interval. Furthermore, since different temporal events may occur concurrently, it would be useful to extract frequent temporal patterns of these events. In this paper the goal is to develop methods that discover temporal arrangements of correlated event intervals which occur frequently in a database.

There are many applications that require mining such temporal relations. One potential application is for analysis of the multiple gestures that occur, in parallel, on the hands and on the face and upper body, to express linguistic information. In signed languages, lexical information is expressed primarily through movements of the hands and arms, whereas critical grammatical information is expressed non-manually, through such behaviors as raised or lowered eyebrows, modifications in eye aperture or gaze, repeated head gestures (nods, shakes) or head tilt, as well as expressions of the nose or mouth. For example, the canonical marking of a wh-question (a question containing a word such as 'who', 'what', 'when', 'where', or 'why') includes lowered brows slightly squinted eyes occurring over a predictable domain (either the question sign or the whole clause constituting the question), and there is frequently a slight rapid head-shake co-occurring with the wh-phrase (Neidle & Lee, 2006). An example of a wh-phrase is shown in Figure 1. Although much is known about the linguistic significance of certain non-manual markings carrying critical syntactic information, there are others whose functions remain to be studied and more fully understood. Pattern detection could ultimately contribute to discovery of the significance of some of these non-manual behaviors. The annotated ASL corpus used for this research was produced by linguists as part of the American Sign Language Linguistic Research Project ((Neidle, et al., 2000; Neidle, 2003; Neidle & Lee, 2006)) using SignStream(TM) ((Neidle, et al., 2001; Neidle, 2002a)). The annotations identify start and end times for: the manual ASL signs (represented by English-based glosses), part of speech for those signs, plus grammatical interpretive labels indicating clusters of non-manual expressions that serve to mark particular syntactic functions (such as wh-questions, negation, etc.) as well as the gestures themselves (e.g., raised eyebrows, wrinkled nose, rapid head shake). See (Neidle, 2002b) for further information about the annotation conventions that were used.

Another application is in network monitoring, where the goal is to analyze packet and router logs. Consider Figure 2 for example, which shows two groups of machines communicating with each other via two routers. In this case an event label is the source or destination IP and the event interval corresponds to the duration of the communication between the two machines. Multiple types of events occurring over certain time periods can be stored in a log, and the goal is to detect general temporal relations of these events that with high probability would describe regular patterns in the network, that could be used for prediction and intrusion detection.

Moreover, interval-based events can be identified in the human gene. More

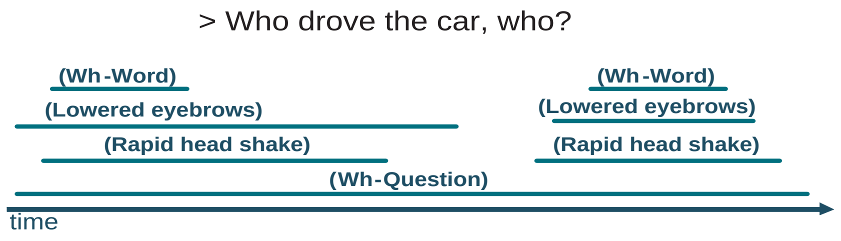


Fig. 1. *An ASL example.*

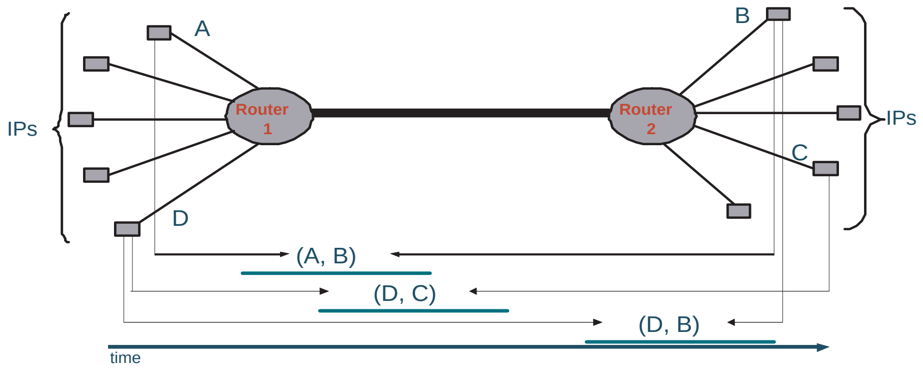


Fig. 2. *A network example.*

specifically, DNA is a sequence of items (nucleotides) defined over a four-letter alphabet, i.e. $\Sigma = \{A, C, G, T\}$. Regions of high occurrence of a nucleotide or combination of nucleotides, known as *poly-regions*, can be defined over DNA. The detection of frequently overlapping poly-regions could lead the biologists to a variety of useful observations concerning the evolution of different genes and their contribution to protein construction. To the best of our knowledge, the first approach to mine frequent arrangements of poly-regions in DNA is discussed in (Papapetrou, et al., 2006).

Most existing sequential pattern mining methods are hampered by the fact that they can only handle instantaneous events, not event intervals. Nonetheless, such algorithms could be retrofitted for the purpose, via converting a database of event intervals to a transactional database, by considering only the start and end points of every event interval. An existing sequential pattern mining algorithm could be applied to the converted database, and the extracted patterns could be post-processed to produce the desired set of frequent arrangements. However, an arrangement of k intervals corresponds to a sequence of length $2k$. Hence, this approach will produce up to 2^{2k} different sequential patterns. Moreover, post-processing will also be costly, since the extracted patterns consist of event start and end points, and for each event interval all the relations with the other event intervals must be determined. Therefore, it is essential to develop interval-based algorithms that can efficiently mine frequent patterns and rules from interval-based data.

The main contributions in this paper include:

- a robust definition of temporal relations between two event intervals that is noise tolerant and through the use of constraints eliminates the ambiguity of Allen's definitions (Allen & Ferguson, 1994),
- a formal definition for the problem of mining frequent temporal arrangements and arrangement rules of event intervals in an event interval database using temporal constraints,
- a prefix-based approach and an efficient algorithm for mining frequent arrangements of temporally correlated events using breadth and depth first search in an enumeration tree of temporal arrangements,
- a further improvement of the mining process with the incorporation of temporal constraints
- an efficient algorithm for mining arrangement rules from the extracted patterns based on user-specified interestingness measures,
- an extensive experimental evaluation of these techniques and a comparison with a standard sequential pattern mining method, SPAM (Ayres et al., 2002), using both real and synthetic data sets.

2. Background

Some basic definitions on temporal logic are presented, followed by a sufficient background on interestingness measures for association rules and finally the problem formulation.

2.1. Ambiguity Issues

Most existing interval-based mining algorithms use Allen's scheme (Allen & Ferguson, 1994) to describe relations between event intervals. Figure 3 shows the twelve main types of relations between two event intervals A and B studied in (Allen & Ferguson, 1994): *before/after, meets/met by, overlaps/overlapped by, starts/started by, during/contains, finishes/finished by.* Because of the limit in the accuracy of demarcating the temporary boundaries of events, there can be variability in these boundaries caused by noisy data. Unfortunately, Allen's relations are hampered by the fact that they cannot capture this variability and thus the representation of a relation might have more than one meaning. This issue has also been addressed in (Moerchen, 2006) and is illustrated in Figure 4. Consider, for instance, the case where the actual relation between two event intervals is $Meets$, but due to noise it appears as *Overlaps* (Figure 4(a)) or $After$ (Figure 4(b)). Similarly, the relation between two *concurrent* events could appear as *Overlaps* (Figure 4(c)) or *Contains* (Figure 4(d)), and also, a *Starts* or $Finishes$ could show up as $Contains$ (Figures 4(e), 4(f)). Such errors can occur due to noisy data and may have a negative influence on the extracted patterns.

In this paper we extend Allen's relations to a more robust scheme that introduces a threshold to achieve relaxation on the boundaries of the relations and eliminates the aforementioned ambiguities.

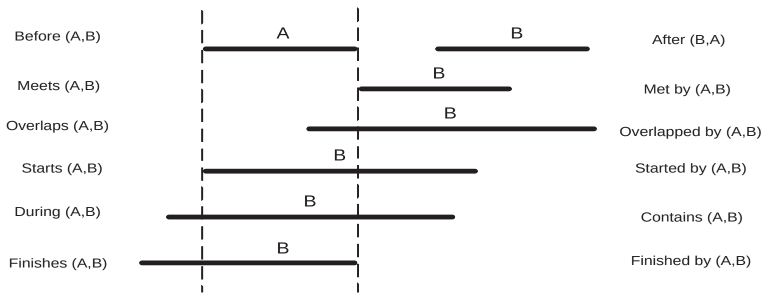


Fig. 3. *Allen's Relations.*

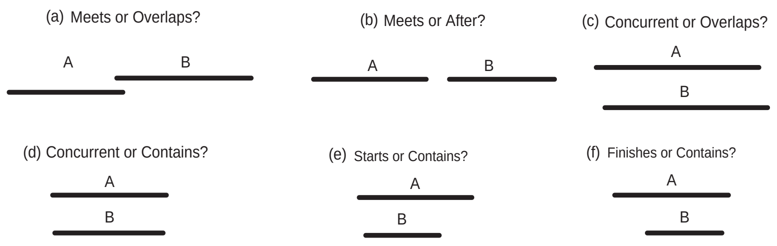

(a) Meets or Overlaps?
A
B
(b) Meets or After?
A
B
(c) Concurrent or Overlaps?
A
B
(d) Concurrent or Contains?
A
B
(e) Starts or Contains?
A
B
(f) Finishes or Contains?
A
B

Fig. 4. *Lack of Robustness in Allen's Relations.*

2.2. Event Interval Temporal Relations

Seven types of temporal relations between two event intervals are considered. Using these relations, general arrangements can be defined. However, the methods presented in this paper are not limited to these relations and can be easily extended to include more types of temporal relations.

Consider two event-intervals A and B, and assume that the user specifies a threshold ϵ to define more flexible matchings between two time intervals. The following relations can be defined (see also Figure 5):

– **Meet(A, B):** B follows A, with B starting at the time A terminates, i.e. $t_e(A) = t_s(B) \pm \epsilon$. This case is denoted as $A \sim B$ and we say that A *meets* B.
– **Match(A, B):** A and B are parallel, beginning and ending at the same time, i.e. $t_s(A) = t_s(B) \pm \epsilon$ and $t_e(A) = t_e(B) \pm \epsilon$. This case is denoted as $A||B$ and we say that A *matches* B.
– **Overlap(A, B):** the start time of B occurs after the start time of A, and A terminates after the start time of B and before B ends, i.e. $t_s(A) < t_s(B)$, $t_e(A) < t_e(B), t_s(B) < t_e(A)$. In addition we need to eliminate any ambiguities that can be caused by the ϵ threshold applied to the other definitions: $t_e(B) - t_e(A) > \epsilon$ and $t_s(B) - t_s(A) > \epsilon$. This case is denoted as $A|B$ and we say that A *overlaps* B.
– **Left-Contain(A, B):** A and B start at the same time and A terminates after B, i.e. $t_s(A) = t_s(B) \pm \epsilon$, $t_e(A) > t_e(B)$ and $t_e(A) - t_e(B) > \epsilon$. This case is denoted as $A \mid > B$ and we say that A *$left$-contains* B.
– **Right-Contain(A, B):** A and B end at the same time and the start time of A precedes that of B, i.e. $t_s(A) < t_s(B)$, $t_e(A) = t_e(B) \ \pm \epsilon$ and and

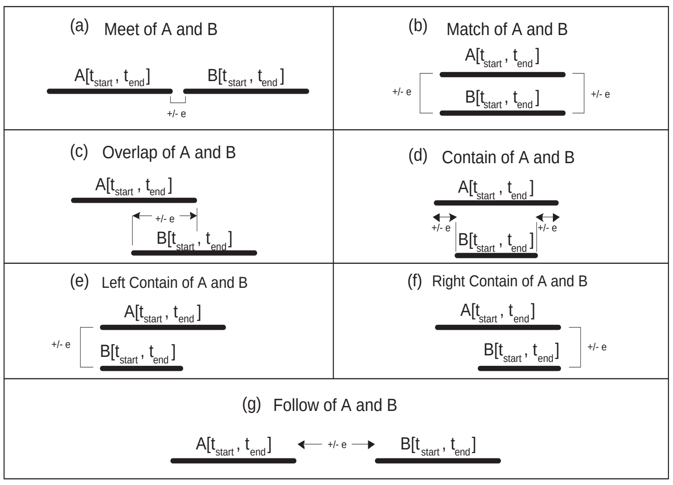


Fig. 5. *Basic relations between two event-intervals: (a) Meet, (b) Match, (c) Overlap, (d) Contain, (e) Left-Contain, (f) Right-Contain, (g) Follow.*

$t_s(B) - t_s(A) > \epsilon$. This case is denoted as $A >| B$ and we say that A *right-contains* B.

- **Contain(A, B):** the start time of B follows the start time of A and the termination of A occurs after the termination of B, i.e. $t_s(A) < t_s(B)$, $t_e(A) > t_e(B)$ and the ϵ threshold is satisfied accordingly: $t_s(B) - t_s(A) > \epsilon$ and $t_e(A) - t_e(B) > \epsilon$. Notice that the ϵ threshold has been applied accordingly to eliminate ambiguities between *contains*, *left-contains*, *right-contains* and *macthes*. This case is denoted as $A > B$ and we say that A *contains* B.
- **Follow(A, B):** B occurs after A terminates, i.e. $t_e(A) < t_s(B)$ and $t_s(B) - t_e(A) > \epsilon$. This case is denoted as $A \rightarrow B$ and we say that B $follows$ A.

2.3. Robustness

By adding ϵ to our interval relations definition all the aforementioned relations are mutually exclusive and at the same time noisy intervals can be handled efficiently. In some applications, however, the user may not want to consider some of the above relations as mutually exclusive. Table 1 shows how these relations cannot be mutually exclusive. For example, a *match* could also be counted as a *left-contain*, *right-contain*, *contain* and/or *overlap*. Also, a *left-contain* or *right-contain* could be counted as a *contain* or an *overlap* as well. Finally, a *meet* could also be counted as a *follow* or *overlap*. Thus, depending on the application, a user might desire to: (1) collapse some relations, e.g. count *left-contain* and *right-contain* as *contain*, or count each *meet* as *follow*, etc., (2) count them multiple times, e.g. each *overlap* is also counted as *left-contain* and *right-contain*, or each *match* is also counted as *contain*, or each *meet* is also counted as *follow*, etc.

Thus, the user has flexibility with respect to which of these options get chosen and clearly it would be application specific.

Table 1. Subsets of Event Interval Relations.

Relation	**Could also be counted as**
meet	follow
	overlap
match	left-contain
	right-contain
	contain
	overlap
left/right-contain	contain
	overlap

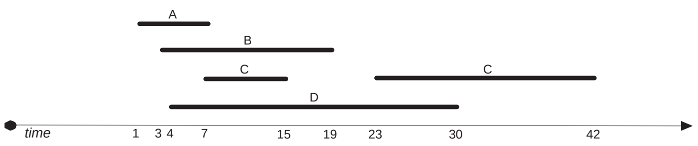


Fig. 6. *An Example of an e-sequence.*

2.4. Arrangements and Arrangement Rules

Let $\mathcal{E} = \{E_1, E_2, ..., E_m\}$ be an ordered set of event intervals, called *event interval sequence* or *e-sequence*. Each E_i is a triple $(e_i,\ t^i_{start}, t^i_{end})$, where e_i is an event label, t^i_{start} is the event start time and t^i_{end} is the end time. The event intervals are ordered by the start time. If an occurrence of e_i is instantaneous, then $t^i_{start} = t^i_{end}$. An e-sequence of size k is called a *k-e-sequence*. For example, let us consider the 5-e-sequence shown in Figure 6. In this case the e-sequence can be represented as follows: $\mathcal{E} = \{(A,1,7),(B,3,19),(D,4,30),(C,7,15),(C,23,42)\}$. Finally, an *e-sequence database* $D = \{\mathcal{E}_1, \mathcal{E}_2, ..., \mathcal{E}_n\}$ is a set of e-sequences.

In an e-sequence database there may be patterns of temporally correlated events; such patterns are called *arrangements*. The definitions given in Section 2.2 can describe temporal relations between two event intervals but they are insufficient for relations between more than two. Consider for example the two cases in Figure 7. Case (a) can be easily expressed using the current notation as: $A|B \rightarrow C$. This is sufficient to determine that A overlaps with B, C follows B and C follows A. On the other hand, the expression for case (b), i.e. $A|B > C$, is insufficient, since it gives no information about the relation between A and C. Thus, we need to add one more operand to express this relation concisely. In order to define an arrangement of more than two events we need to clearly specify the temporal relations between every pair of its events. This can be done by using the "AND" operand denoted by $\star$. Therefore, the above example can be expressed as follows: $A|B \star A|C \star B > C$. Based on the previous analysis, the relations between interval-based events handled in this paper can be expressed using the set of operands: $\mathcal{R} = \{|, ||, >, |>, >|, \sim, \rightarrow\}$ and $\star$.

Consequently, an arrangement $\mathcal{A}$ of n events is defined as $\mathcal{A} = \{\mathcal{E},\ R\}$, where $\mathcal{E}$ is the set of event intervals that occur in $\mathcal{A}$, with $|\mathcal{E}| = n$, and $R = \{R\ (E_1, E_2),$ $R\ (E_1, E_3), ...,\ R\ (E_1, E_n),\ R\ (E_2, E_3), ...,\ R\ (E_2, E_n), ...,\ R\ (E_{n-1}, E_n)\}$. R is the set of temporal relations between each pair (E_i, E_j), for $i = 1,\ ...,\ n$ and $j = i+1,\ ...\ , n-1$, and $R\ (E_i, E_j) \in \mathcal{R}$ defines the temporal relation between E_i and E_j. The size of an arrangement $\mathcal{A} = \{\mathcal{E},\ R\}$ is $|\mathcal{E}|$. An arrangement of size t is called a *t-arrangement*. For example, consider arrangement S' of size 3 shown in

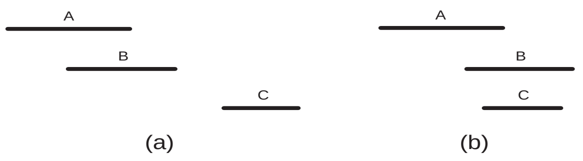


Fig. 7. *(a) S' can be expressed with four operands and (b) S'' cannot.*

Figure 7 (a). In this case $\mathcal{E} = \{A, B, C\}$ and $R = \{R(A,B) = |,\ R(A,C) = \rightarrow,\ R(B,C) = \rightarrow\}$. Given an arrangement $\mathcal{A} = \{\mathcal{E},\ R\}$, a *sub-arrangement* $\mathcal{A}_j$ is an arrangement defined from $\mathcal{A}$ as $\mathcal{A}_j = \{\mathcal{E}_j,\ R_j\}$, where $\mathcal{E}_j \subseteq \mathcal{E}$ and $\mathcal{R}_j \subseteq \mathcal{R}$. Notice that $\mathcal{R}_j$ includes all the relations between the event labels in $\mathcal{E}_j$ that also exist in $\mathcal{R}$. The *absolute support* of an arrangement in an e-sequence database is the number of e-sequences in the database that contain the arrangement. The *relative support* of an arrangement is the percentage of e-sequences in the database that contain the arrangement. Given an e-sequence s, s *contains* an arrangement $\mathcal{A} = \{\mathcal{E}, R\}$, if all the events in $\mathcal{A}$ also appear in s with the same relations between them, as defined in R. Consider again arrangement S' in Figure 7(a) and e-sequence s in Figure 6. We can see that all the event intervals in S' appear in s and further, they are similarly correlated, i.e. Overlap (A,B), Follow (B,C), Follow (A,C). Thus, S' is contained in or *supported* by s. Given a minimum support threshold *min_sup*, an arrangement is *frequent* in an e-sequence database, if it occurs in at least *min_sup* e-sequences in the database.

Based on previous work on itemset and sequence association rules ((Srikant & Agrawal, 1996; Agrawal & Srikant, 1994; Harms, et al., 2002), association rules for arrangements can be defined. Given two arrangements $\mathcal{A}_i$ and $\mathcal{A}_j$ that have been mined from an e-sequence database D, $r : \mathcal{A}_i \Rightarrow^{R_{ij}}_{\lambda,\ D} \mathcal{A}_j$ defines an *arrangement rule* between $\mathcal{A}_i$ and $\mathcal{A}_j$, based on an *interestingness measure* λ. Consider an arrangement $\mathcal{A} = \{\mathcal{E},\ R\}$ that is frequent in D, with $\mathcal{E} = \{e_1,\ ...,\ e_{|\mathcal{E}|}\}$. Let e_l be an event interval in $\mathcal{E}$. $\mathcal{A}$ can be broken into two arrangements $\mathcal{A}_i = \{\mathcal{E}_i,\ R_i\}$, $\mathcal{A}_j = \{\mathcal{E}_j,\ R_j\}$, with $\mathcal{E}_i = \{e_1,\ ...,\ e_l\}$ and $\mathcal{E}_j = \{e_{l+1},\ ...,\ e_{|\mathcal{E}|}\}$; and define a rule between them. Note that $\mathcal{E}$ is split into two sets $\mathcal{E}_i$ and $\mathcal{E}_j$, whereas R_i and R_j are defined based on R, and describe the temporal relations between the event intervals in $\mathcal{E}_i$ and $\mathcal{E}_j$ respectively. Also, R_{ij} defines the set of relations of the event labels $\mathcal{E}_i$ with those in $\mathcal{E}_j$.

2.5. Interestingness Measures

The use of interestingness measures, also known as quantitative measures, plays a very important role in the interpretation of the discovered arrangement rules. Many interestingness measures have been proposed and studied (Kamber & Shinghal, 1996; Hilderman & Hamilton, 2001; Tan, et al., 2002), each of them capturing different characteristics. In this section we give a brief overview of the most common quantitative measures and show how they can be used for mining arrangement rules.

Given a rule $\mathcal{A} \Rightarrow^{R_{AB}}_{\lambda,\ D} \mathcal{B}$, two significant properties of interestingness measures are: *monotonicity* and *anti-monotonicity* (Agrawal & Srikant, 1994):

1. **Monotonicity of an interestingness measure λ:** An interestingness mea-

sure λ is monotone, if for any two arrangements $\mathcal{A}$ and $\mathcal{B}$ (with $\mathcal{A} \subseteq \mathcal{B}$), $\lambda(\mathcal{A}) \leq \lambda(\mathcal{B})$.

2. **Anti-monotonicity of an interestingness measure λ:** An interestingness measure λ is anti-monotone, if for any two arrangements $\mathcal{A}$ and $\mathcal{B}$ (with $\mathcal{A} \subseteq \mathcal{B}$), $\lambda(\mathcal{B}) \leq \lambda(\mathcal{A})$.

Given an arrangement rule: $r : \mathcal{A} \Rightarrow^{R_{AB}}_{\lambda,\, D} \mathcal{B}$, we define *cover*$(\mathcal{A})$ to be the number of e-sequences in D that contain arrangement $\mathcal{A}$ over the size of the e-sequence database D, and *coverage*(r) to be the cover of the antecedent arrangement $\mathcal{A}$. In this paper, we focus on two anti-monotone interestingness measures: (1) support (both for an arrangement and an arrangement rule), (2) all-confidence, and four non anti-monotone: (1) confidence, (2) leverage, (3) lift, and (4) conviction.

2.5.1. Anti-monotone Interestingness Measures

Next, the definitions of two anti-monotone measures are given with respect to an arrangement $\mathcal{A}$ and an event interval database D. Due to the anti-monotonicity property, these measures can be applied on each node and can be used for efficient pruning. Thus given two arrangements $\mathcal{A}$ and $\mathcal{B}$, and an arrangement rule $r : \mathcal{A} \Rightarrow^{R_{AB}}_{\lambda,\, D} \mathcal{B}$:

- $supp(\mathcal{A}) = cover(\mathcal{A})$
  This is the most common quantitative measure among the frequent pattern mining algorithms (Agrawal, et al., 1993). An arrangement with high support guarantees high co-occurrence of its event intervals in D and can produce interesting rules whose antecedent and consequent arrangements are frequent in D. Notice that in our methods we use the standard support counting framework that has been used in the literature for the case where the database consists of a set of sequences.
- $supp(r) = cover(\mathcal{A} \cup \mathcal{B})$
  Similar to the definition of *support* for association rules (Agrawal et al., 1993), the support of an arrangement rule is the number of e-sequences in the database that contain both the antecedent arrangement $\mathcal{A}$ and the consequent arrangement $\mathcal{B}$ of the rule.
- $all\text{-}confidence(\mathcal{A}) = \frac{supp(\mathcal{A})}{max_{1 \leq i \leq t}\{supp(\mathcal{A}_i)\}}$
  Based on the traditional definition of all-confidence (Omiecinski, 2003), the denominator is the maximum number of e-sequences in D that contain any sub-arrangement of $\mathcal{A}$ and t is the size of arrangement $\mathcal{A}$. This states that all-confidence is in fact the smallest confidence of any rule inferred from $\mathcal{A}$.

2.5.2. Non Anti-monotone Interestingness Measures

There has been a great number of interestingness measures proposed and studied, that are not anti monotone. In this paper we consider four of them. Next, we give their definitions with respect to an arrangement rule r implied from an arrangement $\mathcal{A}$ that has been mined from an event interval database D. Note, that since these measures are not anti-monotone, they cannot be used for early pruning during the mining process. Thus, given an arrangement rule $r\ :\ \mathcal{A}\ \Rightarrow^{R_{AB}}_{\lambda,\ D}\ \mathcal{B}$, we have:

- $confidence(r)\ =\ \frac{supp(r)}{coverage(r)}$
  The confidence of a rule (Agrawal et al., 1993) typically expresses the conditional probability of the occurrence of the consequent $\mathcal{B}$ in an e-sequence in D, given that the antecedent $\mathcal{A}$ also occurs in the e-sequence.
- $leverage(r)\ =\ supp(r)\ -\ supp(\mathcal{A}) \times supp(\mathcal{B})$
  Leverage (Piatetsky-Shapiro, 1991) measures the difference between the observed joint frequency of $\mathcal{A}$ and $\mathcal{B}$ (i.e. support of r), and their expected frequency if they were independent. Some useful bounds on leverage have been introduced in (Webb & Zhang, 2005) and are used by the mining algorithms to efficiently prune the search space.
- $lift(r)\ =\ \frac{supp(r)}{supp(\mathcal{A}) \times supp(\mathcal{B})}$
  Lift is a traditional association rule measure (Brin, et al., 1997), and it is the ratio of the observed joint frequency of $\mathcal{A}$ and $\mathcal{B}$, and the expected frequency if they were independent. The problem with this measure is the following: a rule with high lift, may be of little interest since it may have low coverage, meaning that it applies to very few e-sequences of D. At the same time, a rule with low lift might be interesting. In particular, since $coverage(r)\ =\ supp(\mathcal{A})$, we have $\frac{supp(r)}{supp(\mathcal{A}) \times supp(\mathcal{B})}\ =\ \frac{supp(r)}{coverage(r) \times supp(\mathcal{B})}$, and as $coverage(r)\ \downarrow,\ lift(r) \uparrow$.
- $conviction(r)\ =\ \frac{1-supp(\mathcal{B})}{1-confidence(r)}$
  Conviction (Brin, et al., 2004) basically compares the probability of $\mathcal{A}$ appearing without $\mathcal{B}$, assuming independence, with the actual frequency of the appearance of $\mathcal{A}$ without $\mathcal{B}$. A very useful property of conviction is that it is monotone in confidence and lift, i.e.:

$$
\begin{aligned}
conviction(r) &= \frac{supp(\mathcal{A}) \times supp(\overline{\mathcal{B}})}{supp(\overline{\mathcal{B}}, \mathcal{A})} = \frac{supp(\mathcal{A}) \times (1 - supp(\mathcal{B}))}{supp(\mathcal{A}) \times supp(\overline{\mathcal{B}}|\mathcal{A})} \\
&= \frac{1 - supp(\mathcal{B})}{1 - supp(\mathcal{B}|\mathcal{A})} = \frac{1 - supp(\mathcal{B})}{1 - confidence(r)} \\
&= \frac{\frac{1}{supp(\mathcal{B})} - \frac{supp(\mathcal{B})}{supp(\mathcal{B})}}{\frac{1}{supp(\mathcal{B})} - \frac{confidence(r)}{supp(\mathcal{B})}} = \frac{\frac{1}{supp(\mathcal{B})} - 1}{\frac{1}{supp(\mathcal{B})} - lift(r)}
\end{aligned}
$$

Clearly, as $lift(r)\ \uparrow,\ conviction(r) \uparrow$.

2.6. Temporal Constraints

Frequency does not always imply interestingness. A pattern can occur frequently in the database but may not hold interesting information to every user. In addition to the support threshold, the user can also specify a set of temporal constraints $\mathcal{C_T}$ including:

- **A gap constraint** C_g**:** two event intervals that take part in a *follow* relation should be separated by at most C_g time units.
- **A pair of overlap constraints** $C_o =\ \{C_o^l,\ C_o^u\}$**:** the overlap of two event

intervals that take part in an *overlap* relation, is limited by C_o. In fact, C_o^l, C_o^u can be seen as the lower and upper bound of an overlap relation. This means that if their overlap is less than $C_o^l\%$ then their relation is considered a *meet*; if their overlap exceeds $C_o^u\%$ then their relation is considered a *left-contain*. Given two event intervals $E_1 = (e_1, t^1_{start}, t^1_{end})$ and $E_2 = (e_2, t^2_{start}, t^2_{end})$, their *overlap* is equal to $t^1_{end} - t^2_{start}$, if $t^1_{start} < t^2_{start} < t^1_{end}$, otherwise it is zero; and the *overlap percentage* is:

$$overlap\_percentage = \frac{overlap}{min\{t^1_{end} - t^1_{start},\ t^2_{end} - t^2_{start}\}}. \qquad (1)$$

- **A pair of contain constraints** $C_{ct} = \{C^l_{ct}, C^u_{ct}\}$**:** two event intervals that take part in a *contain*, *left-contain* or *right-contain* relation, should have an overlap of at most $C^u_{ct}\%$. If their overlap exceeds this bound, their relation is considered a *match*, whereas if it is less than C^l_{ct} it is discarded.
- **A duration constraint** C_d**:** each event interval should have a duration of at most C_d units. If not, it is discarded.

The set of constraints $\mathcal{C_T}$ is applied during the frequent arrangement extraction.

2.7. Problem Formulation

Based on the above definitions we can now formulate the problem of *constraint-based mining of frequent arrangements of temporal intervals* as follows:

Problem I: Given an e-sequence database D, a set of temporal constraints $\mathcal{C_T}$, and a support threshold *min_sup*, our task is to find set $F = \{\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_n\}$, where $\mathcal{A}_i$ is a frequent arrangement in D and satisfies the constraints in $\mathcal{C_T}$.

We can further extend the previous formulation to extract arrangement rules given an interestingness measure λ. Incorporating interestingness measures and the aforementioned constraints we can formulate the problem of *constraint-based mining of the top-K interesting association rules* as follows:

Problem II: Given a set $\{D, \mathcal{C_T}, \lambda, K, min\_sup\}$, where D is an e-sequence database, $\mathcal{C_T}$ is a set of constraints, λ is an interestingness measure, K is an integer and *min_sup* is the minimum support threshold that implies frequency, we want to mine the top K frequent arrangement rules that satisfy $\mathcal{C_T}$ and maximize λ.

3. Related Work

In this Section, we present the existing work on sequential and temporal pattern mining along with a brief overview of the existing interestingness measures that can be applied during the mining process.

3.1. Sequential Pattern Mining

The first family of sequential pattern mining algorithms are the *Apriori-based* algorithms and their main characteristic is that they apply the *Apriori principle* (Agrawal & Srikant, 1994). The problem of sequential pattern mining was introduced in (Agrawal & Srikant, 1995), along with three Apriori-based algorithms (AprioriAll, AprioriSome and DynamicSome). At each step k, a set of candidate frequent sequences C_k of size k is generated by performing a self-join on $\mathcal{F}_{k-1}$. Notice that $\mathcal{F}_k$ contains all those sequences in C_k of size k that satisfy a user-specified support threshold. The efficiency of support counting is improved by employing a *hash-tree* structure. A more efficient approach, GSP (Generalized Sequential Patterns), was developed in (Srikant & Agrawal, 1996), where time and window constraints are pushed into the mining process. At the same time, (Mannila, et al., 1995) introduced the idea of mining frequent episodes, i.e. frequent sequential patterns in a single long input sequence, using a sliding window to cut the input sequence into smaller segments, and employing a mining algorithm similar to that of Apriori. Notice, however, that in our formulation we focus on finding frequent patterns across a set of input sequences (that constitute a sequence database) and not across a single sequence.

Discovering all frequent sequential patterns in large databases is a very challenging task since the search space is large. Consider for instance the case of a database with m attributes. If we are interested in finding all the frequent sequences of length k, there are $\mathrm{O}(m^k)$ potentially frequent ones. Increasing the number of objects might definitely lead to a paramount computational cost. Apriori-based algorithms employ a bottom-up search, enumerating every single frequent sequence. This implies that in order to produce a frequent sequence of length l, all 2^l subsequences have to be generated. It can be easily deduced that this exponential complexity is limiting all the Apriori-based algorithms to discover only short patterns. A faster and more efficient candidate generation can be achieved using a tree-like structure (*set-enumeration tree*) (Bayardo, 1998) and traversing it in a depth-first search manner to enumerate all the candidate patterns applying efficient pruning techniques. The idea was initially introduced for mining frequent itemsets, but was extended for sequential patterns. SPAM (Ayres et al., 2002), employs a *sequence enumeration tree* to generate all the candidate frequent sequences given the set of event labels. The *root node* of the tree is empty, and each level l contains the complete set of sequences of size l (with each *node* representing one sequence) that can occur in the database. The nodes of each level are generated from the nodes of the previous level and all candidate sequences are enumerated by traversing the tree using depth-first search. For efficient support counting, a bitmap representation of the database is used.

Another family of sequential pattern mining algorithms employ a lattice structure (OXFORD), 2002) for efficient sequence enumeration. The main characteristics of SPADE (Zaki, 2001) include: (1) a vertical representation of the database using *id-lists*, where each pattern is associated with a list of database sequences in which it occurs; all frequent sequences can be enumerated via temporal joins on the id-lists, (2) a lattice-based approach to decompose the original search space into smaller subspaces, which can be processed independently in main memory, (3) within each sub-lattice, two different search strategies (breadth-first and depth-first search) are used for enumerating the frequent sequences. An extension of SPADE, cSPADE (Zaki, 2000), allowed a set of constraints to be placed on the mined sequences. GO-SPADE (Leleu et al., 2003) introduced the idea of *generalized occurrences*. The intuition behind GO-SPADE is that in a sequence database certain items can occur in a consecutive way, i.e. they may appear in consecutive itemsets in the same sequence. To reduce the cost of the mining process, GO-SPADE mainly tries to compact all these consecutive occurrences. Another class of sequential pattern mining algorithms includes the prefix-based ones (Pei et al., 2001; Wang & Han, 2004; Yan et al., 2003). In this case, the database is projected with respect to a frequent prefix sequence and based on the outcome of the projection, new frequent prefixes are identified and used for further projections until the support threshold constraint is violated. A novel tree structure is presented in (Leung, et al., 2007) for incremental pattern mining. The tree captures the content of the original database and can efficiently update itself when there is a change in the database content (insertions, deletions, updates).

Ignoring slight differences in the problem definition, the vast majority of the former algorithms aim at the discovery of frequent sequential patterns based on only a support threshold, which imposes a lack of *user-controlled focus* on the shape of the pattern during the mining process that may sometimes lead to an overwhelming volume of potentially useless patterns. The family of SPIRIT algorithms (Garofalakis et al., 1999) solves this problem by pushing a set of syntactic constraints into the mining process along with a support threshold.

Further studies and works have presented convincing arguments that only closed frequent sequences should be mined targeting more compact results and higher efficiency (Zaki & Hsiao, 2002; Pei, et al., 2000; Wang & Han, 2004; Yan et al., 2003; Pasquier, et al., 1999). Two of the most efficient algorithms for mining frequent closed sequences BIDE (Wang & Han, 2004) and CloSpan (Yan et al., 2003) are based on the notion of the *projected database* and use special techniques to limit the number of frequent sequences and finally only keep the closed ones. CloSpan follows the *candidate maintenance-and-test* approach, i.e. it first generates a set of closed sequence candidates which is stored in a hash-indexed tree structure and then prunes the search space using *Common Prefix* and *Backward Sub-Pattern pruning* (Yan et al., 2003). The main drawback of CloSpan is the fact that it consumes much memory when there are many closed frequent sequences, since pattern closure checking leads to a huge search space; thus, it does not scale very well with respect to the number of closed sequences. In order to face this weakness, BIDE employs a *BI-Directional Extension* paradigm for mining closed sequences, where a *forward directional extension* is used to grow the prefix patterns and check their closure and a *backward directional extension* is used to both check the closure of a prefix pattern and prune the search space. In overall, it has been shown that BIDE has surprisingly high efficiency, regarding speed (an order of magnitude faster than CloSpan) and scalability with respect to database size. Recently, an efficient algorithm for mining maximal sequences has been developed (Luo & Chung, 2008). This algorithm applies both downward and upward closure properties as well as sampling to achieve faster and more efficient pruning. Last but not least, in ConSGapMiner (Ji, et al., 2007) a prefix-based framework is employed and a set of gap and length constraints are applied during the mining process for efficient pruning. The algorithm targets patterns that occur frequently in one class of sequences and are infrequent in sequences of other classes.

3.2. Temporal Mining and Association Rules

Up to this point, the events have been considered to be instantaneous. There have been several approaches on discovering intervals that occur frequently in a transactional database (Lin, 2003; Lin, 2002). In most cases, however, the intervals are unlabelled and no relations between them are considered. (Villafane, et al., 2000) extends the sequential approach by also including the *contain* relation introduced previously. To efficiently mine the arrangements, it employs a *containment graph* representation that imposes a partial order on the event intervals. In (Giannotti, et al., 2006) temporally annotated sequential patterns are considered: these are mainly sequential patterns where each transition from one event to another has a time duration. A graph-based approach is presented in (Hwang, et al., 2004), where each temporal pattern is represented by a graph. In this case however, only two types of relations are considered (*follow* and *overlap*).

Extending earlier work on mining frequent episodes in a single sequence of events (Mannila et al., 1995; Mannila & Toivonen, 1996), there have been various approaches that consider interval-based events. In (Laxman, et al., 2007), a generalized interval-based framework is proposed along with improved support counting techniques for mining interval-based episodes. Correlations between the interval-based events or any possible association rules however, are not being considered. (Hoeppner, 2001; Mooney & Roddick, 2004; Hoeppner & Klawonn, 2001) employ apriori-based techniques to find temporal patterns that occur frequently in the input event sequence. Along with the frequent patterns, they extract association rules and the latest applies some interestingness measures to evaluate their significance. These measures, however, are not pushed into the mining process; they are applied to the set of frequent patterns after the mining process has been completed. Another approach that considers sequences of interval-based events in a database is discussed in (Kam & Fu, 2000). However, the framework used to represent arrangements is limited to certain forms; thus the method is limited to discover certain patterns. A recent BFS-based approach (Winarko & Roddick, 2007) introduces a maximum gap time constraint that can be used to eliminate insignificant patterns. It also employs a straightforward strategy for discovering arrangement rules based on a *minimum confidence* threshold. The rules are extracted after the completion of the mining process and the only interestingness measure applied is the minimum confidence threshold. No further constraints are employed and there is no generalization of the arrangement rule extraction.

Notice that the aforementioned approaches are Apriori-based and do not consider any temporal or structural constraints for the extracted arrangements. Furthermore, the event interval relations used are not robust and cannot efficiently handle noisy data, i.e. noise at the start and end-points of the intervals. To the best of our knowledge, the first *tree-based* approach was proposed in (Papapetrou, et al., 2005), where a *tree-like* structure was used to enumerate the set of arrangements and efficiently mine the frequent ones. In (Wu & Chen, 2007), a non-ambiguous event-interval representation is defined that considers the start and end points of each e-sequence and converts the interval-based representation to a sequential representation. Based on this conversion, a prefix-based algorithm is developed that is similar to (Pei et al., 2001). The problem of this approach is that it cannot scale well as the database size increases, since the proposed representation doubles the size of the database; as a result the number of extracted patterns will be increased in a similar manner as described in Section 1.

In the interim, there has been significant work on discovering association rules on sequential and temporal data. Association rules among items that belong to a frequent itemset are defined in (Srikant & Agrawal, 1996; Agrawal & Srikant, 1994). Similar definitions are given in (Harms et al., 2002) for sequence association rules, and in (Hoeppner, 2001; Hoeppner & Klawonn, 2001) for association rules among interval-based events. In the above works, the evaluation of the rules is achieved by the usage of interestingness measures. The most common ones (introduced in (Agrawal & Srikant, 1994)) are *support* and *confidence*. Using a non Apriori-based technique that avoids multiple database scans, (E.Winarko & J.F.Roddick, 2005) achieved to efficiently mine arrangements and rules in a temporal database. These methods, however, do not consider any constraints for the temporal relations and do not examine any measures for their rules other than the commonly used confidence. Temporal association rules combine traditional association rules with temporal aspects by using time stamps that describe the validity, periodicity, or change of an association. (Oezden, et al., 1998) studies the problem of mining association rules that hold only during certain cyclic time intervals. It is argued that reducing the temporal granularity can lead to the extraction of more interesting rules. In a same fashion, (Chen & Petrounias, 1999; Abraham & Roddick, 1999) consider the discovery of association rules in temporal databases and thus the extraction of temporal features of associated items. The support of the rules is measured only during these intervals. Moreover, in (Ale & Rossi, 2000), the lifetime of an item is defined as the time between the first and the last occurrence and the temporal support is calculated with respect to this interval. In this way, the extracted rules are only active during a certain time, and outdated rules can be pruned by the user. Finally, (Lu, et al., 1998) studies inter-transaction association rules by merging all itemsets within a sliding time window inside a transaction, whereas in (Tsoukatos & Gunopulos, 2001) efficient techniques for mining spatiotemporal patterns are proposed.

3.3. Interestingness Measures

There has been a variety of studies on other interestingness measures (Tan & Kumar, 2000) that provide more accurate results by removing redundancy and limiting the number of extracted rules to the most interesting ones. (Omiecinski, 2003) proposes alternative association rule measures for evaluating the importance of association rules in transactional databases, whereas (Kamber & Shinghal, 1996) introduces some efficient techniques for evaluating the interestingness of rules. An alternative definition of confidence for error-tolerant itemsets and continuous data is described in (Steinbach, et al., 2007). (Hilderman & Hamilton, 1999) carried out a survey on the existing interestingness measures and their significance in association rule mining. In (Hilderman & Hamilton, 2001), a study on the performance of different association rule measures is presented, where different measures are being used to rank the extracted rules and determine the appropriate measure for each dataset. Moreover, (Tan et al., 2002) provides the intuition behind each interestingness measure and gives the basic properties that determine an effective rule measure. (Webb, 2006) proposed generic techniques that provide effective control over the mining process and restrict the number of insignificant rules. Finally, in (Xin, et al., 2006) the mining process is guided by the user's interactive feedback; a *user-specific* interestingness measure is employed,

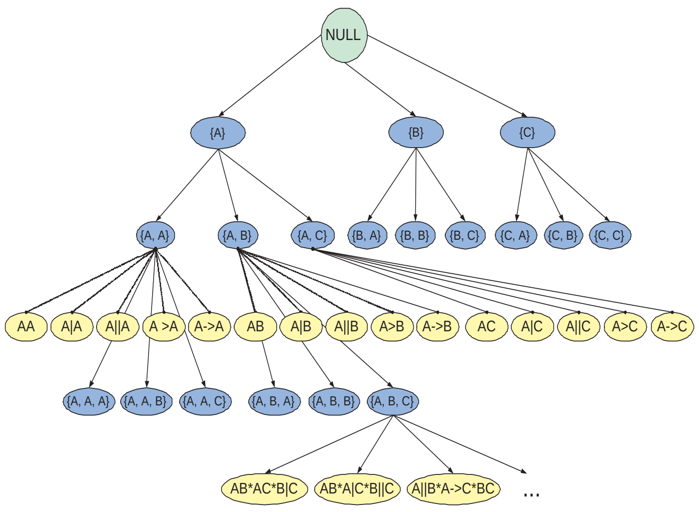


Fig. 8. *An arrangement enumeration tree.*

which consists of a ranking function and a model of prior knowledge that has been defined by the user. Finally, there has been some work on constraint-based mining of frequent itemsets, where the goal is to mine the top K patterns that maximize an interestingness measure (other than the typical support threshold) and satisfy a set of constraints (Webb & Zhang, 2005).

Despite all the aforementioned studies there has been yet no approach that considers interestingness measures on interval-based rules other than the traditional support and confidence.

To recap, there have been various approaches on mining frequent arrangements of temporal intervals; most of them however, are Apriori-based, in some cases (Kam & Fu, 2000) the extracted patterns are limited to certain forms, and no constraints are considered. Furthermore, in most cases the extraction of arrangement rules is performed after the detection of the frequent patterns and no attempt has been made to push it into the mining process. Also, no other measure is used, except for the traditional support and confidence, to evaluate the interestingness of each rule. Current algorithms target all rules that satisfy the desired measures and do not incorporate any constraints regarding the form of each rule. In this work, we present a constrained-based approach that employs a "tree-like" structure to mine frequent arrangements of temporal intervals. Furthermore, the problem of extracting arrangement rules is being considered, and in our case, efficient pruning techniques are applied and the notion of arrangement rules is generalized by including constraints and other interestingness measures except for the traditional support and confidence.

4. Algorithms

A straightforward approach to mine frequent patterns from a database of e-sequences D is to reduce the problem to a sequential pattern mining problem by converting D to a transactional database D'. Without any loss of information, we can keep only the start and end time of each event interval. For example, for every event interval $(e_i,\ t_s,\ t_e)$ in D, that describes an event e_i starting at t_s and ending at t_e, we only keep t_s and t_e in D'. Now, we can apply an efficient existing sequential pattern mining algorithm, e.g., SPAM (Ayres et al., 2002), to generate the set of frequent sequences FS in D'. Every pattern in FS should be post-processed to be converted to an arrangement. However, this approach has two basic drawbacks, regarding cost and efficiency: (1) the patterns in FS will carry lots of redundant information due to the nature of the sequential pattern mining algorithm. In particular, for each extracted frequent pattern, all its sub-patterns will be included in FS. Some of these sub-patterns however, will hold incomplete arrangements and thus FS will include redundant results. Consider for example the following pattern: $f = \{A_{start},\ B_{start},\ A_{end},\ C_{start},\ C_{end},\ B_{end}\}$, and assume that it is frequent in D' and thus it is included in FS. Then the sequential pattern algorithm will also generate all possible sub-patterns of f, e.g. $f_1 = \{A_{start},\ B_{start}\}$, $f_2 = \{A_{start},\ B_{start},\ A_{end}\}$, $f_3 = \{A_{start},\ B_{start},\ B_{end}\}$, etc. These sub-patterns correspond to incomplete arrangements and they constitute redundancy for the result set. As shown in Section 5 for small supports a sequential pattern mining approach can yield up to 70% redundancy; (2) post-processing can be costly, since each frequent pattern f should be converted to an arrangement. For the arrangement representation scheme presented in this paper, the complexity for post-processing f is $O(|f|^2)$, since we need to define all pair-wise relations between each event interval in the sequence. This cost can be reduced if more efficient representations are used, however the problem of redundancy remains.

Next, we describe three efficient algorithms for mining frequent arrangements of temporal intervals that address the previous problems. The first two, employ a tree-based enumeration structure, like the one used in (Bayardo, 1998; Zaki, 2001; Ayres et al., 2002). The first algorithm uses BFS to generate the candidate arrangements, whereas the second uses DFS. Although the BFS-based approach is equivalent to Apriori and (Winarko & Roddick, 2007), the algorithm is further extended to include temporal and structural constraints. The third algorithm employs a prefix-growth approach, similar to (Pei et al., 2001). In our approach the interval-based representation of the database is retained as opposed to (Wu & Chen, 2007); thus we achieve a better scalability.

4.1. The Arrangement Enumeration Tree

The tree-based structure used by the first two algorithms is called *arrangement enumeration tree*. An arrangement enumeration tree is shown in Figure 8. The *root* of the tree contains the empty set. Each level k consists of a set of *nodes*, denoted as $N(k)$, that hold the complete set of k-arrangements. Let n_i^k denote node i on level k, where i indicates the position of n_i^k in the k-th level based on the type of traversal used by the algorithm. Every node $n_i^k \in N(k)$ holds a set of labels $\mathcal{E}$. These labels are used to create all the arrangements that can be defined using those labels and all the available types of relations. These arrangements are enumerated using a set of intermediate nodes $M^k(n_i^k)$. Each node in $M^k(n_i^k)$ is linked directly to n_i^k and represents an arrangement $\mathcal{A}$=$\{\mathcal{E}, R\}$ where $\mathcal{E}$ is defined by the labels in n_i^k and R is one of all the possible combinations of relations for the labels in $\mathcal{E}$. For the case shown in Figure 8, $\mathcal{E} = \{A,\ B,\ C\}$ and on level 1, $N(1) = \{\{A\}, \{B\}, \{C\}\}$, i.e. we have one node for every item in $\mathcal{E}$. Then, performing temporal joins

A	
esid	*Intv-List*
1	[1, 3]
1	[6, 12]
2	[1, 2]
2	[10, 12]
3	[4, 7]
3	[9, 11]
4	[6, 14]

B	
esid	*Intv-List*
1	[1, 3]
1	[8, 11]
2	[2, 6]
2	[11, 15]
3	[1, 3]
3	[11, 12]
4	[1, 5]
4	[7, 10]

Fig. 9. *ISId-Lists for items A and B.*

on the nodes of level 1, the set of the 2-arrangements of Level 2 is generated, with $N(2) = \{\{A,\ A\},\ \{A,\ B\},\ \{A,\ C\},\ \{B,\ A\},\ \{B,\ B\},\ \{B,\ C\},$ $\{C,\ A\},\ \{C,\ B\},\ \{C,\ C\}\}$, and for each node n_i^2 set $M^k(n_i^k)$ is defined. In general, on level k: (1) $N(k)$ is created by joining the nodes in $N(k\text{-}1)$ with those in $N(1)$, (2) for every node n_i^k, $M^k(n_i^k)$ is defined and then linked to n_i^k. The arrangement enumeration tree is created as described above, using the set of operands defined in Section 2 and it is traversed using either breadth-first or depth-first search.

4.2. BFS-based Approach

In this section we consider an event interval mining algorithm that uses the arrangement enumeration tree described above to generate the set of candidate arrangements and then prunes those that are not frequent or cannot lead to any frequent arrangement if expanded. The algorithm traverses the tree using breadth first search which is equivalent to the Apriori-based approaches described in Section 3. The main characteristic of this algorithm is that a set of constraints has been incorporated into the mining process.

4.2.1. The Mining Process

To accelerate the mining process, the *ISIdLists* (Interval Sequence Id Lists) structure is introduced, that attains a compact representation of the intervals and a relatively low join cost. More specifically, an ISIdList is defined for every arrangement generated by this process. The head of the list is the representation of the arrangements using $\mathcal{R}$ and the event labels comprised in it; each e-sequence is of type (*id*, *intv-List*), where *id* is the e-sequence id in D that supports the arrangement, and *intv-List* (interval List) is a double-linked list of all the time intervals during which the arrangement occurs in the corresponding e-sequence in D.

Consider, for example, an e-sequence database D with three unique items A, B and C, as in Figure 10. The ISIdLists of A and B are shown in Figure 9. Let $\mathcal{F}_k$ denote the complete set of frequent k-arrangements and C_k the set of candidate frequent k-arrangements. Our algorithm will first scan D to find $\mathcal{F}_1$, i.e. the complete set of 1-arrangements. To achieve this, a scan will be performed on D for every event type e_i. If the number of e-sequences in D that contain an interval of e_i satisfies the support threshold, e_i will be added to $\mathcal{F}_1$, and its ISIdList will be updated accordingly.

In order to generate the candidate 2-arrangements, we use the arrangement

Database D	
id	*e-sequence*
1	A [1, 3], B [1, 3], A [6, 12], B [8, 11], C [9, 10]
2	A [1, 2], B [2, 6], A [10, 12], B [11, 15], C [14, 17]
3	B [1, 3], A [4, 7], A [9, 11], B [11, 12] , C [12, 14]
4	B [1, 5], A [6, 14], B [7, 10], C [8, 9]

Fig. 10. *An e-sequence database D.*

enumeration tree described above to get the nodes of level 2, along with the set of their corresponding intermediate nodes. These nodes are generated based on $\mathcal{F}_1$ found in the previous step. The way this is done in (Papapetrou et al., 2005) is too naive: all possible combinations of 2 event intervals would be checked by making multiple scans (one per candidate 2-arrangement) on the ISIdLists of each event label; the ones satisfying the minimum support threshold constraint would be added to $\mathcal{F}_2$. In this paper we follow a more efficient approach which is based on the Apriori principle: for any 2-arrangement that is frequent in D, each of the two event intervals that form the arrangement should occur in at least min *_sup* number of transactions. Thus, once we get the event labels of the frequent 1-arrangements, we check which pairs of these event labels co-occur in at least min *_sup* transactions. If a pair of event labels does not satisfy the above criteria, then there is not need to check the intermediate nodes for any potential frequent temporal relations. This technique introduces additional pruning and is applied at each candidate generation step, resulting in a significant acceleration of the mining process, as shown in the experimental section.

Moving to the next levels, i.e. generating the set of frequent k-arrangements, we traverse the nodes on level k-1. Note that these nodes correspond to the set of frequent (k-1)-arrangements. For every node n_i^{k-1}, a new node n_i^k is created on level k. The aforementioned pruning technique is also applied here: given a node on level k-1 and a new event label, we check whether the number of transactions where they co-occur is above min *_sup*. If so, the set of intermediate nodes $M^k(n_i^k)$ is generated, one for every type of correlation of the items in n_i^k; otherwise that node is pruned. For every node in $M^k(n_i^k)$ an ISIdList is created that contains: (1) the set of items of n_i^k, (2) the types of 2-relations between them, (3) for every type of 2-relation a pointer to the intermediate nodes on Level 2 that correspond to that 2-relation. Also, note that if an arrangement is found to be infrequent, then the node in the tree that corresponds to that arrangement is no further expanded.

For efficient support counting we adopt a "bitmap-like" representation (Ayres et al., 2002). Each ISIdList contains an array $\mathcal{T}$ of size $|D|$, where $\mathcal{T}(j) = 1$ if the arrangement represented by the ISIdList is contained in e-sequence j, and $\mathcal{T}(j) = 0$ otherwise. When two ISIdLists are joined, a logic AND is performed on the two bitmap arrays and the bitmap array for the new node is determined. The support of each arrangement is determined by summing its bitmap array.

The above process is more clear through the following example: consider database D in Figure 10 and assume that $min\_sup = 2$. Scanning D and filtering with min_sup, we get $F_1 = \{\{A\}, \{B\}, \{C\}\}$. Based on F_1 and the enumeration tree, set F_2 of the frequent 2-arrangements is generated. In our case, we get all the possible pairs of the 1-arrangements in F_1, i.e. N(2), and for every pair of events in the arrangements, D is scanned to get all the types of relations between them, i.e.

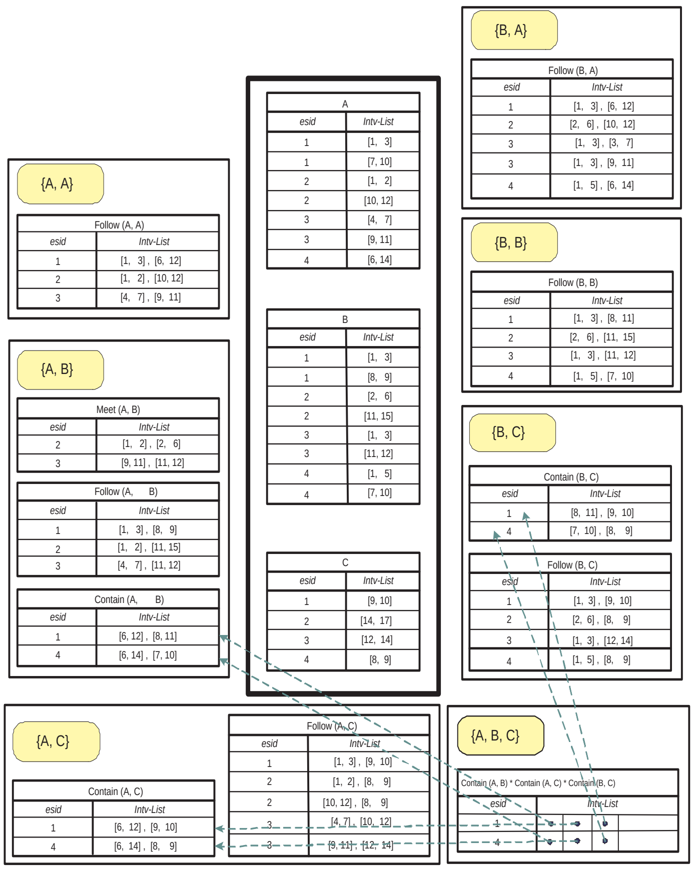


Fig. 11. *The set of frequent 2 and 3-arrangements.*

M^2. If these relations satisfy the support threshold they are added to F_2. Then we produce F_3 based on F_2. The algorithm first creates $N(3)$, following a breadth-first search traversal, along with the set of intermediate nodes. Every node in M^3 that satisfies *min_sup* is added to F_3, which in our case consists of only one arrangement: $\{(A,\ B,\ C),\ (>,\ \ >,\ \ >)\}$. F_1, F_2 and F_3 are shown in Figure 11. The main steps of this method are described in Algorithm I, considering an input database D, a minimum support threshold *min_sup*, an interestingness measure λ, a set of constraints $\mathcal{C}$ and an integer k.

4.2.2. Applying Constraints

The most naive way of applying the set of constraints $\mathcal{C}_{\mathcal{T}}$ is to do so after the generation of each arrangement and before the application of any interestingness measure. Applying constraints, however, is meant to speed up the mining process and the degree to which $\mathcal{C}_{\mathcal{T}}$ is applied for pruning determines the efficiency of pruning. Gap, overlap and contain constraints are applied during the second step of the algorithm, when the set of frequent 2-arrangements is created. Finally,

```
input   : $D$: a database of e-sequences.
          $min\_sup$: minimum support threshold.
          $\mathcal{C}_\mathcal{T}$: a set of temporal constraints.
          $\lambda$: an interestingness measure.
          $K$: an integer.
          $\epsilon$: the event interval threshold.

output  : The set $F$ of the frequent arrangements in $D$ that satisfy $\mathcal{C}_\mathcal{T}$.
          The set $A_R$ of the top $K$ rules that satisfy the constraint $\lambda$.
$F = \emptyset$;
$C_1 = \emptyset$;
foreach event type $e_i$ do
    if $e_i$ exists in $D$ then
        $C_1 = C_1 \cup e_i$;
    end
end
$F_1 = \{e_i \in C_1 \mid e_i.cupport \geq min\_sup$ , $\mathcal{C}_d$ is satisfied$\}$; $j = 2$; while $F_{j-1} \neq \emptyset$ do
    $N(j) = generate\_candidates\ (N(j-1), N(1))$;
    // The next set of nodes on the tree is determined, following BFS traversal
    foreach node $n_i^j \in N(j)$ do
        $M^j(n_i^j) = generate\_krelations(j, \mathcal{C}_\mathcal{T})$;
        // this function generates the nodes in $M^j$, along with their ISIdLists.
        // for the case where $j = 2$, it ensures that $\mathcal{C}_{ct}$, $\mathcal{C}_g$ and $\mathcal{C}_o$ are satisfied.
        $C_j = M^j$;
        foreach candidate $c \in C_j$ do
            if $c.support < min\_sup$ then
                $C_j.remove(c)$; // removes $c$ from $C_j$.
                $prune\_subtree(c)$; // prunes subtree($c$).
            end
        end
        $F_j = C_j$;
        $extract\_rules(D, C_j, \lambda, K)$; // this is Algorithm II
    end
end
```

Algorithm I: *A BFS-based algorithm for discovering the complete set of frequent temporal arrangements and the top K arrangement rules in a database of e-sequences given a set of constraints and an interestingness measure.*

the duration constraint is applied at the first step when the set of frequent 1-arrangements is created.

4.2.3. Generating Arrangement Rules

Regarding the rule generation, two approaches can be followed: one is to extract the rules after the mining process is completed, i.e. given the complete set F of frequent arrangements and the user-specified interestingness measure λ, we apply a technique similar to the one proposed in (Agrawal & Srikant, 1994) to extract the rules implied from F and maximize λ. The second and more efficient approach is to prune using λ *with optimistic estimates. This means that during the mining process we are going to use* λ *for pruning as aggressively as possible.* This, however, depends on whether λ satisfies the anti-monotonicity property.

If so then it can be incorporated into the mining process; if not then the first approach is employed. More details on this step are given in Section 4.6.

4.3. DFS-based Approach

In the BFS approach the arrangement enumeration tree is explored in a top-bottom manner, i.e. all the children of a node are processed before moving to the next level. On the other hand, when using a depth-first search approach, all sub-arrangements on a path must be explored before moving to the next one. A DFS approach for mining frequent sequences has been proposed in (Tsoukatos & Gunopulos, 2001). Based on this, the previous algorithm can be easily modified to use a depth-first search candidate generation method. This can be done by adjusting function *generate_candidates*() so that it follows a depth-first search traversal. Consider the previous example: our algorithm will first generate node $n_1^1 = \{A\}$ followed by $M(n_1^1)$, then $n_1^2 = \{A,\ A\}$ followed by $M(n_1^2)$, and so on. Again, the constraints are applied in a similar fashion as in BFS; the rule extraction is described in Section 4.6.

The advantage of DFS over BFS is that DFS can very quickly reach large frequent arrangements and therefore, some expansions in the other paths in the tree can be avoided. For example, say that a k-arrangement $\mathcal{A}$ is found to be frequent. Then, the set of all sub-arrangements of $\mathcal{A}$ will also be frequent according to the Apriori principle. Thus, those expansions can be skipped, reducing the cost of computation. To do so, one more step is added to Algorithm 4.2.1: when a node is found to contain a frequent arrangement, each sub-arrangement is added to F. However, this approach has a significant drawback: in BFS each node in M^k for $k \geq 3$ contains pointers to its corresponding nodes in M^2. In DFS this is not possible since M^k is not completely generated due to the type of traversal. Therefore, in the current case, DFS needs to scan through D multiple times to determine the frequency of each 2-relation contained in a candidate arrangement. On the other hand, BFS already had this information available in M^2. Knowing the set of 2-arrangements before constructing the set of 3-arrangements can prevent us from making expansions that will lead to infrequent arrangements.

4.4. Hybrid DFS-based Approach

A hybrid event interval mining approach is considered, based on the following observation: since the ISIdLists contain pointers to the nodes on the second Level of the tree, a DFS approach would be inappropriate since for every node n_i^k we would have to scan the database multiple times to detect the set of 2-relations among the items in that node. In the BFS approach these nodes will already be available, since they have been generated in the second step of the algorithm. Thus, we use a hybrid DFS approach that generates the first two levels of the tree using BFS and then follows DFS for the rest of the tree. This would compensate for the multiple database scans discussed above, since the set of frequent 2-arrangements will already be available thereby eliminating the need for multiple database scans.

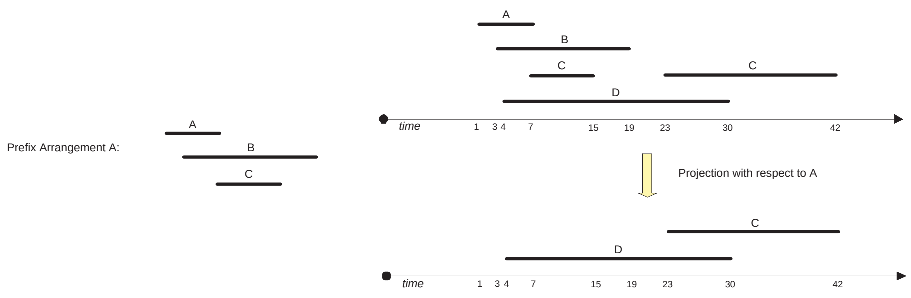


Fig. 12. *An Example of a Projection.*

4.5. A Prefix-based Approach

A prefix-based algorithm for mining frequent arrangements of temporal intervals is presented. We will show that in the case of interval-based events, a prefix-growth approach is quite inefficient, especially when the size of the e-sequences is large and there is repetition of the same event labels in the same e-sequence.

Consider an arrangement $\mathcal{A} = \{\mathcal{E}, R\}$ and an e-sequence S. The *projection of* S with respect to $\mathcal{A}$ is the remaining part S' of S, if the *first occurrence* of $\mathcal{A}$ in S is removed. Figure 12 shows an example of a projection. Next, we define the projection of an e-sequences database with respect to an arrangement. Using the definition given in (Pei et al., 2001) for the sequential approach, we can define the *projection of an e-sequence database* D with respect to an arrangement $\mathcal{A}$ as the e-sequence database D' produced from D, if each e-sequence in D is projected with respect to $\mathcal{A}$. However, this definition is incomplete. The problem is illustrated by the example shown in Figure 13, where an e-sequence database of two e-sequences is considered. Following the basic steps of the prefix-growth mining algorithms with support threshold $min\_sup = 2$, we have:

1. Scan D for frequent 1-arrangements: in our case we detect $\mathcal{A}$ and $\mathcal{C}$.
2. Project the database with respect to each of the arrangements found at Step 1.
3. The projection with respect to $\mathcal{A}$ is shown in Figure 13 and will yield one new locally frequent arrangement, $\mathcal{C}$, since the support threshold equal to 2.
4. The result of Step 3 is the detection of $A \rightarrow C$ in the first e-sequence and $A \mid C$ in the second.
5. Another projection follows with respect to $\mathcal{C}$, but it produces an empty e-sequence database and therefore the mining process is terminated.

As it can be seen, we failed to get $A \mid C$ with support 2. In fact, $A \mid C$ was produced after the first projection, as shown in Figure 13, but it was not considered frequent since its support was erroneously calculated as 1. This example shows that when an e-sequence database is projected with respect to a prefix arrangement, finding only the first occurrence of the arrangement may hide some patterns and prevent the mining algorithm from detecting them.

Thus, given an e-sequence database D and an arrangement $\mathcal{A}$, the *projected e-sequence database* D' with respect to $\mathcal{A}$ can be obtained from D, if from each e-sequence in D we find *every* occurrence (not just the first one) of $\mathcal{A}$ and project with respect to each one of them. It can be seen that such an approach can lead

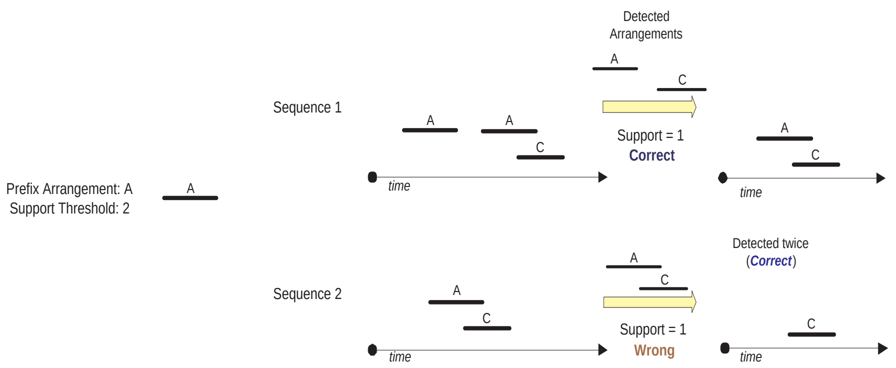


Fig. 13. *An Example of an e-sequence database of two e-sequences and a projection that does not work.*

to a huge computational cost, since for each e-sequence in the database, all the combinations of the occurrences of a prefix should be examined and not just the first one.

In our experimental evaluation we show the performance of the prefix-growth approach and compare it with the BFS and DFS approaches presented previously.

4.6. Applying Other Interestingness Measures

In the previous sections, three efficient methods are presented for mining the complete set F of frequent arrangements of an e-sequence database D. What remains to be done is to discover the set of top K arrangement rules that maximize the given interestingness measure λ in D. A very important issue here, is how optimistic can our pruning be using λ. This, in fact, depends on the properties of λ and more specifically on anti-monotonicity. Next, we present two approaches to handle λ based on whether it is anti-monotone or not.

4.6.1. Handling Interestingness Measures that do not Preserve Anti-monotonicity

If an interestingness measure λ does not preserve the anti-monotonicity property, we have two options: (1) infer the arrangement rules from the extracted frequent arrangements after the mining process is completed (this process is similar to the one described in (Agrawal & Srikant, 1994) for mining association rules), (2) find an upper or lower bound for λ and apply pruning with optimistic estimates.

The first option is quite straightforward. Given F and D, for each $\mathcal{A}_i \in F$, with $\mathcal{A}_i = \{\mathcal{E}, R\}$:

1. $\mathcal{E}$ is split into two sets $\mathcal{E}_1$ and $\mathcal{E}_2$, such that $\mathcal{E}_1 \cup \mathcal{E}_2 = \mathcal{E}$ and $\mathcal{E}_1 \cap \mathcal{E}_2 = NULL$. Based on R, two arrangements are defined: $\mathcal{A}_1 = \{\mathcal{E}_1, R_1\}$ and $\mathcal{A}_2 = \{\mathcal{E}_2, R_2\}$.
2. Apply λ on $r : \mathcal{A}_1 \Rightarrow^{R_{12}}_{\lambda,\ D} \mathcal{A}_2$.
3. If r satisfies λ add it into the set of valid rules, else discard r.

4. If the set of rules has reached the desired size K, the rule with the smallest λ value (let it be λ_min) is removed and replaced by the new rule, as long as the new rule's value is greater than λ_min. If not, then the new rule is discarded.

The above algorithm will produce the complete set of top K arrangement rules that maximize λ. Moreover, based on the mining process and the constraints applied during the extraction of the frequent patterns we have ensured that these rules will satisfy the set of constraints $\mathcal{C}_T$.

However, it would be more efficient if λ could guarantee early and optimistic pruning during the mining process. The only problem is that λ is not anti-monotone. One way to overcome this issue and achieve some pruning is to find a bound value (upper or lower) $bound\_\lambda$ for λ, such that when an arrangement $\mathcal{A}$ is reached on the tree, if $\lambda(\mathcal{A}) < Upperbound\_\lambda < \lambda$, then none of the rules implied by $\mathcal{A}$ can lead to an arrangement that satisfies λ and thus these rules can be pruned. In a similar fashion, if $\lambda(\mathcal{A}) > Lowerbound\_\lambda > \lambda$, then all rules implied by $\mathcal{A}$ can lead to an arrangement that satisfies λ; thus all these rules are immediately added to the result. One such bound was used in (Bayardo, et al., 1999) for association rules and the following claim extends it for the case of arrangement rules:

Claim 1: If $all\text{-}confidence(\mathcal{A}) \geq \lambda$, then all rules implied by $\mathcal{A}$ can lead to an arrangement that satisfies $confidence$. Thus these rules are not considered any further and are immediately added to the result set.

Proof: Since $all\text{-}confidence$ of an arrangement is the minimum confidence of any rule inferred from it, the Claim is straightforward.

Another sort of pruning would be to find a way to imply whether a rule satisfies λ by calculating a simpler form of the rule that reduces the computational cost. There have been several works on bounding interestingness measures for frequent itemset mining (Bayardo et al., 1999; Webb & Zhang, 2005; Omiecinski, 2003). These bounds were used to prune the search space during the rule extraction process. In this paper we borrow some of those bounds and infer some new ones to incorporate some of the non anti-monotone measures into the mining process.

1. **Bounding Confidence:**
   Given an arrangement $\mathcal{A}$, for any rule $r: \ \mathcal{A}_1 \ \Rightarrow^{R_{12}}_{\lambda,\ D} \ \mathcal{A}_2$ implied from $\mathcal{A}$, where λ in this case stands for confidence, we have the following claim:

   Claim 2: If $\frac{cover(\mathcal{A}_2)}{cover(\mathcal{A}_1)} < min\_Confidence$, then r does not satisfy λ.

   Proof: Straightforward from the definition of confidence.

2. **Bounding Leverage:**
   Given an arrangement $\mathcal{A}$, for any rule $r: \ \mathcal{A}_1 \ \Rightarrow^{R_{12}}_{\lambda,\ D} \ \mathcal{A}_2$ implied from $\mathcal{A}$, where λ in this case stands for leverage, we have the following claim:

   Claim 3: If $cover(\mathcal{A}_1) \ > \ 1 - \frac{min\_Leverage}{cover(\mathcal{A}_2)}$ or $cover(\mathcal{A}_2) \ > \ 1 - \frac{min\_Leverage}{cover(\mathcal{A}_1)}$,

```
input   : $D$: a database of e-sequences.
          $C_k$: the set of candidate $k$-arrangements.
          $\lambda$: an interestingness measure.
          $K$: an integer.
output  : The set $A_R$ of top $K$ arrangement rules in $D$ that satisfy $\lambda$.
$A_R = \emptyset$;

foreach $\mathcal{A}_i = \{\mathcal{E}, R\} \in C_k$ do
    if $\lambda$ is anti-monotone and $\mathcal{A}_i$ does not satisfy $\lambda$ then
        prune the subtree beyond $\mathcal{A}_i$;
        continue;
    end
    apply bounds on $\mathcal{A}_i$;
    if $\lambda(\mathcal{A}_i) \leq Upperbound\_\lambda \leq \lambda$ then
        continue;
    end
    // apply lower bound, if such bound exists
    foreach $e \in \mathcal{E}$ do
        split $\mathcal{E}$ into two sets $\mathcal{E}_1 = \{e_1, ..., e\}$ and $\mathcal{E}_2 = \{e+1, ..., e_{|\mathcal{E}|}\}$;
        $prune\_split()$;
        // based on the set of rules $\mathcal{R}_\mathcal{A}$ defined in the previous level
        // prune the split if necessary.
        based on $R$: define $\mathcal{A}_1 = \{\mathcal{E}_1, R_1\}$ and $\mathcal{A}_2 = \{\mathcal{E}_2, R_2\}$;
        apply $\lambda$ on $r: \mathcal{A}_1 \Rightarrow^D_\lambda \mathcal{A}_2$;
        if $r$ does not satisfy $\lambda$ then
            continue;
        end
        if $|A_R| \geq K$ then
            $min$ = minimum $\lambda$-value in $A_R$;
            if $r.\lambda \geq min$ then
                remove rule with $\lambda = min$;
                $A_R = A_R \cup r$;
            end
        end
        if $|A_R| < K$ then
            continue;
        end
    end
end
```

Algorithm II: *Extracting the set of arrangement rules given a set of candidate arrangements.*

then r does not satisfy λ.

Proof: Shown in (Webb & Zhang, 2005) for association rules and it can be easily extended for arrangement rules.

4.6.2. Handling Interestingness Measures that Preserve Anti-monotonicity

In this subsection we consider the case where the interestingness measure λ preserves the anti-monotonicity property. If λ is anti-monotone, it can be used for pruning very early during the mining process. When an arrangement $\mathcal{A}$ is reached on the enumeration tree, if there exists no rule r inferred from $\mathcal{A}$ satisfying λ, then the subtree of the node representing $\mathcal{A}$ is pruned, since it cannot produce any interesting rule. However, if there exists a set of rules $\mathcal{R}_{\mathcal{A}}$ for which λ holds, then the mining process continues with the subtree. In this case though, the rules that are going to be discovered in the subtree depend on $\mathcal{R}_{\mathcal{A}}$, based on which, the rule extraction process can be accomplished faster by excluding those rules that will definitely not satisfy λ. Let $\mathcal{R}_{\mathcal{A}} = \{r_1,\ r_2,\ ..., r_n\}$ be the set of rules inferred from the arrangement at node n on the tree, where $r_i\ :\ \mathcal{A}_i \Rightarrow^{R_{\mathcal{A}_i\mathcal{B}_i}}_{\lambda,\ D}\ \mathcal{B}_i$. When n is expanded, for each new arrangement $\mathcal{C} = \{\mathcal{E},\ R\}$, we follow the same process as in the previous section to discover the new arrangement rules, however the search is limited by $R_{\mathcal{A}}$ as follows:

1. $\mathcal{E}$ is split into two sets $\mathcal{E}_1$ and $\mathcal{E}_2$.
2. If there is no arrangement $\mathcal{A}_i = \{\mathcal{E}_i,\ R_i\}$ in the antecedent part of any rule in $\mathcal{R}_{\mathcal{A}}$, such that $\mathcal{E}_1 \supseteq \mathcal{E}_i$, then due to the anti-monotonicity property, $\mathcal{E}_1$ cannot be the antecedent part of any rule inferred from $\mathcal{C}$; thus this split is skipped.

As it can be seen, an anti-monotone interestingness measure λ can be used for more efficient and *optimistic pruning* (meaning that it can be used to prune as aggressively as possible) and thus lead to a faster rule extraction. Algorithm II shows how we can extract the set of top K rules, given set of frequent arrangements.

5. Experimental Setup

Experiments that compare the performance of our algorithms with SPAM (Ayres et al., 2002) are presented. [1] All experiments have been performed on a 2.8Ghz Intel(R) Pentium(R) 4 dual-processor machine with 2.5 gigabytes main memory, running Linux with kernel 2.4.20. The algorithms have been implemented in C++, compiled using g++ along with the -O3 flag, and their run time has been measured with the output turned off. Note that for SPAM, the post-processing time of converting the sequential patterns to arrangements has not been counted. Also, as mentioned in Section 4, SPAM is tuned as follows: for every event interval we keep only the start and end time; as for the postprocessing phase, the frequent arrangements are extracted from the sequential patterns as described in the same section. The patterns found by SPAM consist of a set of start and end points of event intervals, which are converted to arrangements at the postprocessing phase. SPAM manages to discover the same patterns extracted by our algorithms, but, as expected, it produces a great number of redundant patterns.

For our experimental evaluation we have used both real and synthetic datasets. Next, we present an analysis of our experimental evaluation, first by comparing

[1] The code was obtained from: `http://himalaya-tools.sourceforge.net/Spam/`.

our algorithms with respect to their run time and then by showing their performance for each of the interestingness measures described in Section 2.5.

5.1. Experiments on Real Data

We have performed a series of experiments on two real datasets. One was an annotated database of ASL utterances, which is available online at: `http://www.bu.edu/asllrp/`. The other was a sample network dataset of ODFlows taken from Abilene, which is an Internet2 backbone network, connecting over 200 US universities and peering with research networks in Europe and Asia.

5.1.1. Experiments on the ASL SignStream Database

The first series of experiments have been performed on the ASL database created by the National Center for Sign Language and Gesture Resources at Boston University. The SignStream(TM) database used in this experiment consists of a collection of 884 utterances, where each utterance associates a segment of video with a detailed transcription. Non-manual markings play a crucial role in the grammar of ASL (Baker-Shenk, 1983; Coulter, 1979; Liddell, 1980; Neidle et al., 2000), thus for our experiments we focused only on: specific non-manual gestures (e.g., raised eyebrows, head tilt forward), functional identification of clusters of these non-manual gestures that carry syntactic meaning (e.g., 'wh-question', 'negation'), and part-of-speech identifications of manual signs (e.g., verb, wh-word), each one occurring over a time interval. The overall list of field names and labels included in the database are given in Table 2.

We first tested our algorithms on subsets of sentences from the database: those that contained marking of a wh-question, and another that contained marking of negation. Our goal was to detect all frequent arrangements that occurred during wh-questions and negative sentences. In these two datasets, called *Dataset* 1 and *Dataset* 2 respectively, the number of e-sequences was 73 and 68, with an average number of items per sequence equal to 32 and 26 respectively. Since all four algorithms produce the same results, in our experiments we compare their run time. As shown in Figures 16(a) and 16(b), Hybrid DFS outperformed both BFS and SPAM for supports less than 30%. Note that for a support threshold of 3%, SPAM produced 25628 patterns as opposed to 7574 arrangements produced by Hybrid DFS, which induces 70% redundancy in the results, and for a support threshold of 5% the redundancy was 63.4%. On average, Hybrid DFS was approximately 1.5 to 2 times faster than BFS and almost three times (or more) faster than SPAM. In many cases, the performance of the prefix-growth approach was very poor, as it was predicted in the analysis of Section 4. We tested the qualitative performance with respect to the setting of parameter ϵ, where varied between 2 and 6 instants. In our case, low values of ϵ gave the most meaningful results, since the amount of noise in the ASL dataset was limited to a small number of frames. For higher values of ϵ, the detected relation types changed, and as a result the number of extracted frequent arrangements decreased, hiding many of the interesting patterns. For all experiments presented in this section, $\epsilon = 3$.

Next, our algorithms were tested on the whole Signstream(TM) database

Table 2. List of Field Names and Labels.

Fields	**Field Names**	**Field Labels**
Head position	tilt fr/bk	tilt fr/bk s
	turn	turn
	tilt side	tilt side
	jut	jut
Head movement	nod	nod
	shake	shake
	side to side	side< − >side
	jut	jut
Body	body lean	body lean
	body mvmt	body mvmt
	shoulders	shoulders
Eyes, Nose and Mouth	eye brows	eye brows
	eye gaze	eye gaze
	eye aperture	eye apert
	nose	nose
	mouth	mouth
	English mouthing	English mouthing
	cheeks	cheeks
Neck	neck	neck
Grammatical information	negative	negative
	wh question	wh question
	yes-no question	yes-no question
	rhetorical question	rhq
	topic/focus	topic/focus
	conditional/when	cond/when
	relative clause	rel. clause
	role shift	role shift
	subject agreement	subj agr
	object agreement	obj agr
	adverbial	adv
Part of Speech	POS	POS
	Non-dominant POS	POS2
Gloss Fields	main gloss	main gloss
	non-dominant hand gloss	nd hand gloss
Text Fields	English translation	english

that contained 884 utterances with an average e-sequence size of 29 items per e-sequence. We refer to this as *Dataset* 3. The algorithms have been tested for various supports and have been compared in terms of run time. The experimental results in Figure 16(c) show that in terms of run time, the Hybrid DFS-based approach outperforms the BFS-based especially in small supports. SPAM starts with a run time between that of BFS and Hybrid DFS and for small supports the run time increases dramatically. The prefix-growth approach is again very poor.

The results produced by our algorithms have been examined and evaluated by linguists who had been involved in collecting the ASL data and producing the annotations for this dataset [2]. According to their feedback, our algorithms

[2] Carol Neidle and Robert G. Lee, of Boston University.

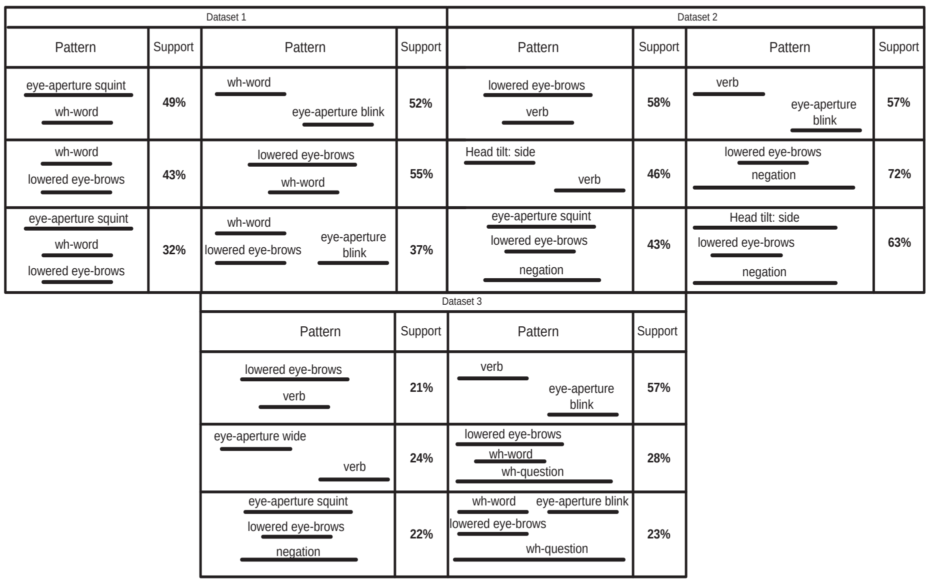


Fig. 14. *Some Frequent Patterns of Datasets* 1, 2 *and* 3.

managed to detect a set of ASL patterns that have been already known: for example, the strong correlation between wh-question marking and lowered eye-brows. Similar correlations were found between the occurrence of what had been labelled as ”negative” marking and the non-manual behaviors that comprise this marking (such as side-to-side head shake). Similarly, it was unsurprising that wh-words co-occurred with the non-manual markings associated with wh-questions. Nonetheless, it is good that our approaches independently found known correlations. Also, some other discovered patterns were considered to be trivial (e.g., that the onset of a behavior preceded the behavior itself) or in some cases, an artifact of the selection criterion used to define the sample of data under consideration. For example, within the set of negative sentences, verbs frequently co-occurred with marking of negation (whereas this would not be true of verbs in non-negative sentences). For example, a “verb” would always occur in a “negation”, or a “wh-word” is always included in a “wh-question”. Linguists have evaluated and confirmed the correctness of the patterns and rules discovered by our algorithms. In Figure 14 we can see some of the most frequent arrangements detected in *Datasets* 1, 2, *and* 3. In terms of memory requirements, Hybrid DFS outperformed the rest, as shown in Figure 15 for Dataset 3. The other two ASL datasets had similar behavior. For all three datasets Hybrid DFS outperformed SPAM by a factor of 2.4 and BFS by a factor of 1.8.

5.1.2. Applying Interestingness Measures on the ASL Datasets

The main motivation behind the application of interestingness measures during the mining process was to reduce the number of extracted patterns to the most interesting ones (for the user), removing most of the trivial cases described previously. All six interestingness measures presented in Section 4 have been applied

Table 3. Number of Extracted Rules from Datasets 1, 2, 3, and 4.

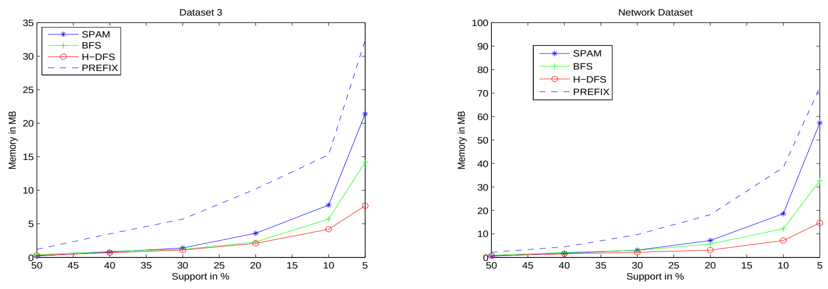
λ	**Supp**	λ	**ds 1**	**ds 2**	**ds 3**	**ds 4**
lift	0.5	0.1	87	22	25	156
lift	0.5	0.3	55	16	9	133
lift	0.5	0.5	46	14	2	87
lift	0.4	0.5	187	149	53	241
lift	0.3	0.5	453	47	155	786
conviction	0.5	0.5	29	47	13	68
conviction	0.4	0.5	112	47	16	142
confidence	0.5	0.5	46	14	2	81
confidence	0.4	0.5	148	53	14	251
leverage	0.4	0.3	6	1	6	25
leverage	0.4	0.2	25	3	12	43
all-conf.	0.5	0.5	46	15	15	84
all-conf.	0.4	0.5	54	53	59	122
all-conf.	0.3	0.5	67	56	64	143

to the ASL Datasets, leading to the discovery of different sets of rules that maximize each one of them. The basic observation from the extracted patterns on the ASL database was that applying only the support threshold yielded a huge number of redundant patterns that do not provide any useful information. Therefore, except for the support we applied all the interestingness measures described previously. Table 3 shows the number of rules extracted by our algorithms for each of the three ASL Datasets. As we can see, lift, confidence and conviction produced an interesting number of rules, whereas the number of rules that maximize leverage and all-confidence was pretty small. In Figure 17 we present some of the top patterns in the ASL dataset with regard to each interestingness measure. Notice that the first column of the figure gives the complete arrangement that takes part in the rule, the second column shows the antecedent arrangement, the third column shows the consequent arrangement and the fourth column gives the value of each interestingness measure.

The application of interestingness measures managed to remove trivial cases and preserve those already known to the linguists. In our case $lift$ and *conviction* had the best performance among all the measures we examined. *Leverage* also did a very good job in removing trivial rules, however it also removed a great number of known rules, and in many cases the number of extracted rules was extremely small. Our experimental evaluation showed that the combination of interestingness measures and the efficient arrangement mining algorithms can potentially provide more meaningful results. Nonetheless, an evaluation by experts is always needed to determine the most effective measure for a given application.

5.1.3. Experiments on Network Data

Our algorithms have also been tested on a network dataset of 960 e-sequences with an average e-sequence size of 100 items per e-sequence. The data has been obtained from a collection of ODFlows obtained from Abilene, that consists of 11 Points of Presence (PoPs), spanning the continental US. Three weeks of sampled IP-level traffic flow data were collected from every PoP in Abilene for the period December 8, 2003 to December 28, 2003. We have selected two routers that were shown to have a high communication rate with each other, and have monitored the IP connections from one (LOSA: router in LA) to the other (ATLA: router in Altanta) for three days. An e-sequence in our dataset is the set of IP connections



(a) (b)

Fig. 15. *Memory requirements for Dataset* 3 *and the network dataset.*

from LOSA to ATLA for every 15 minutes. Due to the huge number of IP addresses, we have selected 200 IPs that appear most frequently in these three days. The dataset that resulted from the above process is called *Dataset* 4.

Our experiments focused only on run time for the same reasons described earlier. In a qualitative experiment, the parameter ϵ was tuned to vary between 3 and 15. Due to the nature of the dataset, the number of extracted patterns was huge since the average e-sequence size was too high. As before, the number and type of extracted patterns varied with respect to ϵ; unfortunately, no interesting or surprising patterns were found. In this case, "interestingness" means that the patterns were meaningful (from a network point of view), and described an expected communication behavior of the two routers. The run time comparison of the four algorithms is shown in Figure 16(d), and it is quite similar to that of the ASL datasets; again Hybrid DFS outperforms the other algorithms in low supports. In terms of memory requirements, the trends are the same as those for the ASL datasets. In Figure 15 we can see that for low *supports*, BFS consumes twice as much memory as Hybrid DFS, whereas the Prefix-based approach requires four times the memory used by Hybrid DFS.

5.1.4. Applying Interestingness Measures on the Network Dataset

As far as the arrangement rules are concerned, all six interestingness measures have been applied and a great number of rules have been extracted, most of which were interesting (from the point of view described earlier). In Table 3 we can see the number of rules generated by each interestingness measure. The main observation here is that the number of rules generated for the network dataset by each measure is greater than that for the ASL datasets. This is expected due to the nature of the network dataset, i.e. the average e-sequence size is larger, and the number of relations (and thus arrangements) is greater, due to the high communication rate in the network.

5.2. Applying Temporal Constraints

The usage of temporal constraints is application specific. Depending on the patterns that the user is targeting and also on the nature of the dataset, an optimal setting can be defined for the temporal constraints. This setting will result in the elimination of undesired patterns and at the same time provide further pruning.

Table 4. Performance of H-DFS with Constraints on Dataset 3. μ is the mean interval duration in the dataset.

Const. type	**Const. value**	**# of extracted patterns**	**Runtime in sec**
Gap	∞	7574	512
Gap	39 ($\mu\times$ 2)	4375	261
Gap	52 ($\mu\times$ 3)	4453	278
Gap	65 ($\mu\times$ 4)	4562	299
Gap	78 ($\mu\times$ 5)	5766	345
Overlap	$[19.5, \infty) = [\mu/2, \infty)$	6175	426
Overlap	$(13, 19.5] = (\mu/3, \mu/2]$	5512	331
Overlap	$(9.75, 13] = (\mu/4, \mu/3]$	5409	315
Overlap	$(0, 9.75] = (0, \mu/4]$	5321	302
Contain	no bounds	7574	512
Contain	(80%, 100%]	6441	484
Contain	(70%, 80%]	6521	492
Contain	(60%, 70%]	6625	501

Temporal constraints have been applied to both real datasets (Dataset 3 and the Network dataset) and achieved efficient pruning. Temporal constraints have been set according to the mean duration of the intervals in each dataset. Table 4 provides a summary of the impact of these constraints on Dataset 3 (the *support* threshold has been set to 3% and the mean interval duration is 19.5 seconds). It can be seen that as constraints get tighter, the number of extracted patterns decreases and at the same time the run time improves. The same behavior has been observed with the Network dataset. In general, the effectiveness of temporal constraints highly depends on the nature of the dataset and the user requirements. If for example the event intervals in a dataset are spread apart then the *gap* constraint should be set to a high value; if on the other hand they are too close to each other then the *gap* constraint should be low. The best way of setting these constraints is to first extract some statistics on the given dataset, i.e. mean, standard deviation, etc., and then consult with the experts to determine what type of constraints and what thresholds should be applied to that specific dataset; this way we can achieve further pruning at a lower run time and at the same guarantee meaningful results.

5.3. Experiments on Synthetic Data

Due to the relatively small size of the current SignStream database, we have generated numerous synthetic datasets to test the efficiency of our algorithms.

5.3.1. Synthetic Data Generation

The following factors have been considered for the generation of the synthetic datasets: (1) number of e-sequences, (2) average e-sequence size, (3) number of distinct items, (4) density of frequent patterns. Using different variations of the above factors we have generated several datasets. In particular, our datasets were of sizes 200, 500, 1000, 2000, 5000 and 10000, with average sequence sizes of 3, 10, 50, 100 and 150 items per e-sequence. Moreover, we have tried various

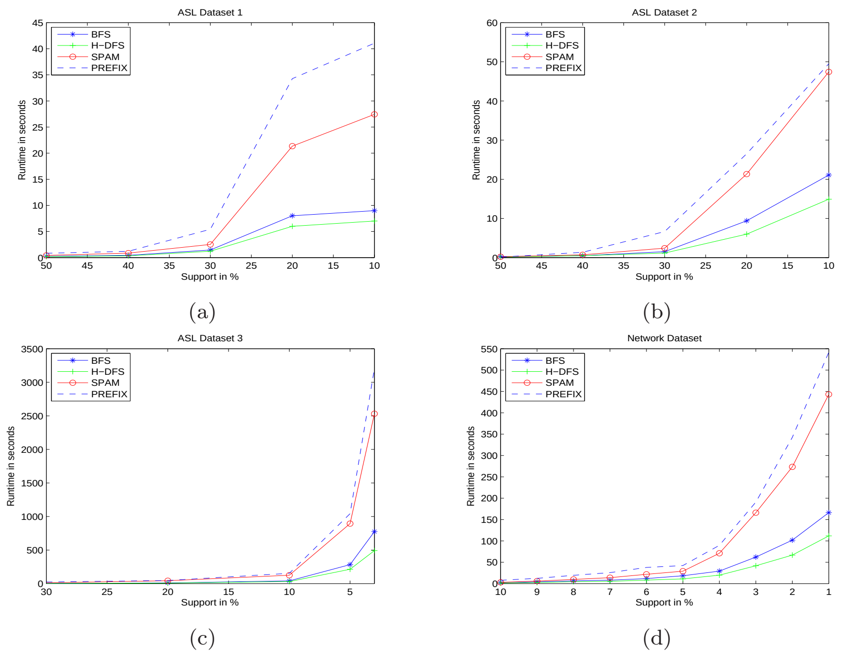
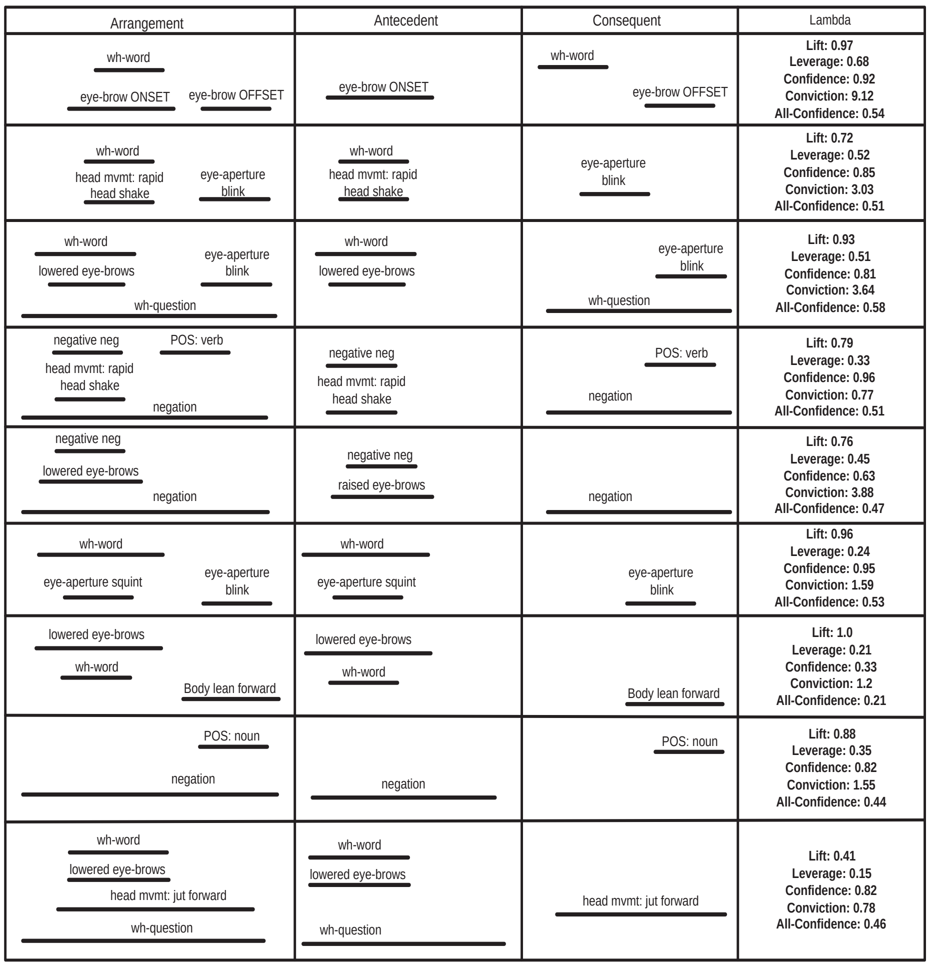


Fig. 16. *Results on Real Datasets: (a) ASL Dataset 1: $|S|$: 73, $|A|$: 52, $|\mathcal{E}|$: 400.; (b) ASL Dataset 2: $|S|$: 68, $|A|$: 26, $|\mathcal{E}|$: 400.; (c) ASL Dataset 3: $|S|$: 884, $|A|$: 102, $|\mathcal{E}|$: 400.; (d) Network Dataset: $|S|$: 960, $|A|$: 100, $|\mathcal{E}|$: 200 (where $|S|$ denotes the size of the dataset, $|A|$ the average sequence size), and $|\mathcal{E}|$ the number of distinct items in the dataset.*

numbers of distinct items, i.e. 400, 600 and 800. Also, we have considered different densities of frequent patterns. We first created a certain number of frequent patterns that with medium support thresholds of 20% (sparse), 40% (medium density) and 60% (dense) would generate a lot of frequent patterns and then added random event intervals on the generated sequences.

5.3.2. Experimental Results

The experimental results have shown that Hybrid DFS clearly outperforms BFS, and especially in low support values and large database sizes Hybrid DFS is twice as fast as BFS. Regarding the performance of SPAM, we have concluded that in medium support values and small database sizes SPAM performs better than BFS but worse than Hybrid DFS, whereas in small support values and large datasets BFS outperforms SPAM. We compared the four algorithms on several small, medium and large datasets for various support values. The results of these tests are shown in Figure 20. As expected, SPAM performs poorly in large sequences and small supports. This behavior is expected since for every arrangement produced by BFS and Hybrid DFS, SPAM generates all the possible subsets of the start and end points of the events in that arrangement. As the database size grows along with the average e-sequence size, SPAM will be producing a great number of redundant frequent patterns that yield to a rapid increase of its run time. In all cases, the prefix-growth algorithm performs very poorly. In

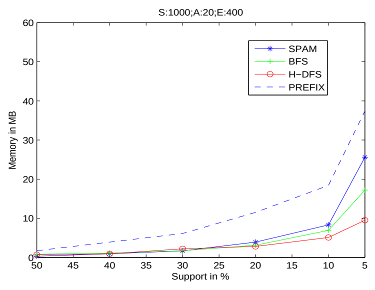


Arrangement	Antecedent	Consequent	Lambda
wh-word eye-brow ONSET eye-brow OFFSET	eye-brow ONSET	wh-word eye-brow OFFSET	Lift: 0.97 Leverage: 0.68 Confidence: 0.92 Conviction: 9.12 All-Confidence: 0.54
wh-word head mvmt: rapid head shake eye-aperture blink	wh-word head mvmt: rapid head shake	eye-aperture blink	Lift: 0.72 Leverage: 0.52 Confidence: 0.85 Conviction: 3.03 All-Confidence: 0.51
wh-word lowered eye-brows eye-aperture blink wh-question	wh-word lowered eye-brows	eye-aperture blink wh-question	Lift: 0.93 Leverage: 0.51 Confidence: 0.81 Conviction: 3.64 All-Confidence: 0.58
negative neg POS: verb head mvmt: rapid head shake negation	negative neg head mvmt: rapid head shake	POS: verb negation	Lift: 0.79 Leverage: 0.33 Confidence: 0.96 Conviction: 0.77 All-Confidence: 0.51
negative neg lowered eye-brows negation	negative neg raised eye-brows	negation	Lift: 0.76 Leverage: 0.45 Confidence: 0.63 Conviction: 3.88 All-Confidence: 0.47
wh-word eye-aperture squint eye-aperture blink	wh-word eye-aperture squint	eye-aperture blink	Lift: 0.96 Leverage: 0.24 Confidence: 0.95 Conviction: 1.59 All-Confidence: 0.53
lowered eye-brows wh-word Body lean forward	lowered eye-brows wh-word	Body lean forward	Lift: 1.0 Leverage: 0.21 Confidence: 0.33 Conviction: 1.2 All-Confidence: 0.21
POS: noun negation	negation	POS: noun	Lift: 0.88 Leverage: 0.35 Confidence: 0.82 Conviction: 1.55 All-Confidence: 0.44
wh-word lowered eye-brows head mvmt: jut forward wh-question	wh-word lowered eye-brows wh-question	head mvmt: jut forward	Lift: 0.41 Leverage: 0.15 Confidence: 0.82 Conviction: 0.78 All-Confidence: 0.46

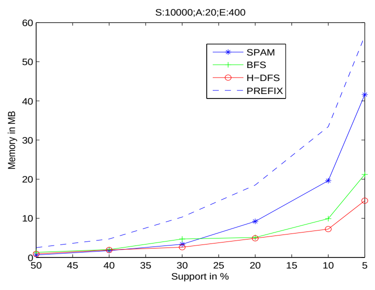
Fig. 17. *Some of the discovered rules in Dataset 1 and Dataset 2.*

medium supports and small datasets it can still do better than SPAM, but in smaller supports and larger datasets its performance decreases dramatically.

In terms of memory requirements, Hybrid DFS outperformed SPAM by a factor of 2.6 and BFS by a factor of 2.1. Some results are shown in Figure 18. For all the other cases (different e-sequence sizes, number of items per e-sequence and average number of items per e-sequence) the trends are similar and thus further results are omitted. It can be seen that despite the fact that SPAM needs only one integer to represent an item of a sequence whereas BFS and Hybrid DFS need three integers (one for the item, one for the start and one for the end point), the later still outperform by an order of magnitude. This proves the inefficiency of SPAM on interval-based event sequences. Also, the efficiency of our algorithms has been tested with respect to different number of items and different e-sequence size. The performance of the four algorithms is shown in Figure 19. It can be seen that Hybrid DFS outperforms BFS by

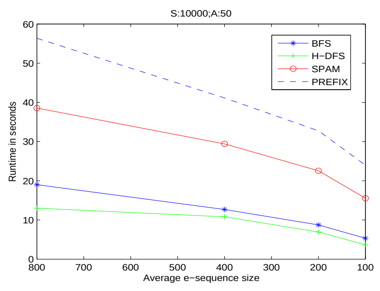


(a)

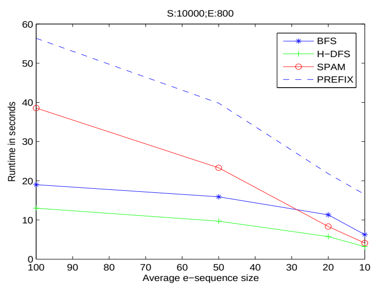


(b)

Fig. 18. *Memory requirements for the synthetic dataasets:(a) 1000 e-sequences with* 20 *items per e-sequence on average, and* 400 *event labels; (b) 10000 e-sequences with* 20 *items per e-sequence on average, and* 400 *event labels.*



(a)



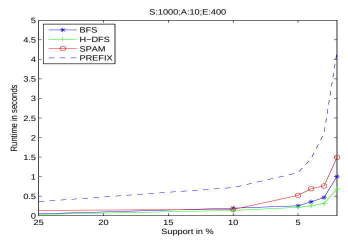
(b)

Fig. 19. *Runtime comparison with respect to different e-sequence size and different average number of items per e-sequence:(a) 10000 e-sequences with* 50 *items per e-sequence on average; (b) 10000 e-sequences with* 400 *event labels.*

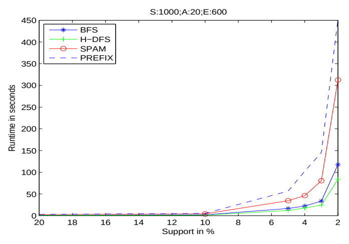
at least an order of magnitude. SPAM and the Prefix-based approach perform much worse, as expected.

6. Conclusions

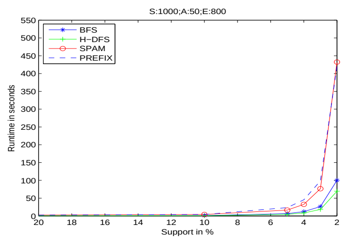
In this paper, we have formally defined the problem of constraint-based mining of frequent temporal arrangements of event interval sequences and presented three efficient methods to solve it. The first two approaches use an arrangement enumeration tree to discover the set of frequent arrangements. The DFS method further improves performance over BFS by reaching longer arrangements faster and hence eliminating the need for examining smaller subsets of these arrangements. The prefix-growth approach is poor in performance, since the number of projections can be really huge, especially when the input e-sequences have repetitions of the same event label. We further extended our algorithms by applying constraints during the mining process. These constraints provide a more user-specified focus on the structure of the extracted patterns. Moreover, except for the support threshold, we have applied other interestingness measures and focused on mining the top k arrangement rules that maximize a given inter-



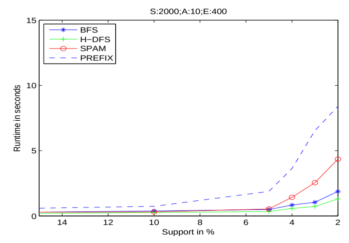
(a)



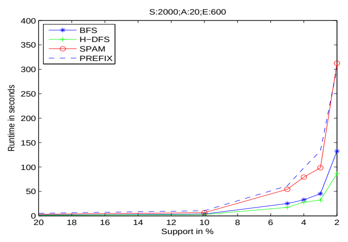
(b)



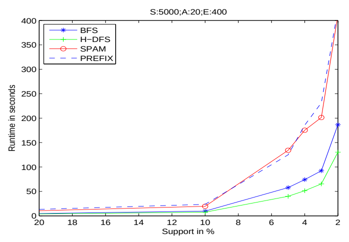
(c)



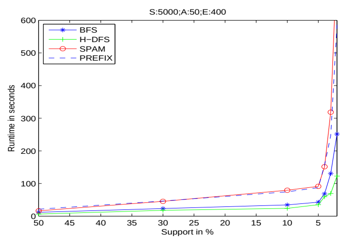
(d)



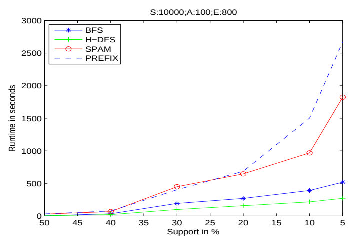
(e)



(f)



(g)



(h)

Fig. 20. *Results on Synthetic Datasets: (a) Dataset* 1*:* $|S|$*:* 1000*,* $|A|$*:* 10*,* $|\mathcal{E}|$*:* 400*, frequent patterns of medium density.; (b) Dataset* 2*:* $|S|$*:* 1000*,* $|A|$*:* 20*,* $|\mathcal{E}|$*:* 600*, sparse frequent patterns.; (c) Dataset* 3*:* $|S|$*:* 1000*,* $|A|$*:* 50*,* $|\mathcal{E}|$*:* 800*, dense frequent patterns.;(d) Dataset* 4*:* $|S|$*:* 2000*,* $|A|$*:* 10*,* $|\mathcal{E}|$*:* 400*, frequent patterns of medium density.;(e) Dataset* 5*:* $|S|$*:* 2000*,* $|A|$*:* 20*,* $|\mathcal{E}|$*:* 400*, frequent patterns of medium density.;(f) Dataset* 6*:* $|S|$*:* 5000*,* $|A|$*:* 20*,* $|\mathcal{E}|$*:* 400*, dense frequent patterns.;(g) Dataset* 7*:* $|S|$*:* 5000*,* $|A|$*:* 50*,* $|\mathcal{E}|$*:* 400*, dense frequent patterns.;(h) Dataset* 8*:* $|S|$*:* 10000*,* $|A|$*:* 100*,* $|\mathcal{E}|$*:* 800*, dense frequent patterns.*

estingness measure. Our experimental evaluation demonstrates the applicability and usefulness of our methods.

An interesting direction for future work is to adjust the definition of support to capture multiple occurrences of an arrangement within a database sequence. Furthermore, we could develop an efficient algorithm for mining closed arrangements. In this case however, a prefix-based approach, like BIDE (Wang & Han, 2004), would be extremely costly. Therefore, we should come up with a method to produce the complete set of closed arrangements that will employ more efficient projections or use different techniques to prevent the paramount cost of multiple projections. Another direction for future work is to mine partial orders of temporal arrangements and closed temporal arrangements. The notion of mining partial orders of sequential patterns has been introduced in (Mannila & Toivonen, 1996) and an interesting approach has been recently proposed for closed sequential patterns in (Casas-Garriga, 2005). However, these methods again assume that the events are instantaneous. Last but not least, our algorithms could be applied on biological data, such as genes of different organisms (Papapetrou et al., 2006). The ultimate goal would be to extract frequent arrangements of nucleotide regions and produce interesting rules. These patterns could be further used to determine various features of different groups of organisms and possibly detect mutations or tandem repeats.

Acknowledgements. This material is based upon work supported in part by the following National Science Foundation Grants: IIS-0705749, IIS-0308213, IIS-0329009 and IIS-0812309. We thank the anonymous reviewers for their very useful comments and suggestions. Also, we would like to thank professor Carol Neidle (Department of Romance Studies, Linguistics Program) for her very useful comments and feedback.

References

T. Abraham & J. F. Roddick (1999). 'Incremental Meta-Mining from Large Temporal Data Sets'. In *ER '98: Proceedings of the Workshops on Data Warehousing and Data Mining*, pp. 41–54.

R. Agrawal, et al. (1993). 'Mining association rules between sets of items in large databases'. In *Proc. of ACM SIGMOD*, pp. 207–216.

R. Agrawal & R. Srikant (1994). 'Fast algorithms for mining association rules'. In *Proc. of VLDB*, pp. 487–499.

R. Agrawal & R. Srikant (1995). 'Mining Sequential Patterns'. In *Proc. of IEEE ICDE*, pp. 3–14.

J. M. Ale & G. H. Rossi (2000). 'An Approach to Discovering Temporal Association Rules'. In *Proc. of the SAC*, pp. 294–300.

J. Allen & G. Ferguson (1994). 'Actions and Events in Interval Temporal Logic'. *Journal of Logic and Computation* .

J. Ayres, et al. (2002). 'Sequential PAttern Mining using a Bitmap Representation'. In *Proc. of ACM SIGKDD*, pp. 429–435.

C. Baker-Shenk (1983). 'A Micro-analysis of the Nonmanual Components of Questions in American Sign Language'. *Doctoral Dissertation* .

R. Bayardo, et al. (1999). 'Constraint-based Rule Mining in Large, Dense Databases'. In *Proc. of IEEE ICDE*, pp. 188–197.

R. J. Bayardo (1998). 'Efficiently mining long patterns from databases'. In *Proc. of ACM SIGMOD*, pp. 85–93.

S. Brin, et al. (1997). 'Beyond marketbaskets: generalizing association rules to correlations'. In *ACM International Conference on Management of Data (SIGMOD)*, p. 265276.

S. Brin, et al. (2004). 'Dynamic itemset counting and implication rules for market basket data'. In *ACM International Conference on Management of Data (SIGMOD)*, p. 255264.

G. Casas-Garriga (2005). 'Summarizing Sequential Data with Closed Partial Orders'. In *Proc. of SDM.*

X. Chen & I. Petrounias (1999). 'Mining Temporal Features in Association Rules'. In *Proc. of PKDD*, pp. 295–300, London, UK. Springer-Verlag.

G. R. Coulter (1979). 'American Sign Language Typology'. *Doctoral Dissertation* .

B. Davey & H. Priestley (2002). *Introduction to Lattices and Oder*. Cambridge University Press.

E.Winarko & J.F.Roddick (2005). 'Discovering Richer Temporal Association Rules from interval-based Data'. In *In Proc. DaWaK*.

M. Garofalakis, et al. (1999). 'SPIRIT: Sequential pattern mining with regular expression constraints'. In *Proc. of VLDB*, pp. 223–234.

F. Giannotti, et al. (2006). 'Efficient Mining of Temporally Annotated Sequences'. In *SDM*.

J. Han, et al. (2000). 'FreeSpan: Frequent pattern-projected sequential pattern mining'. In *Proc. of ACM SIGKDD*, pp. 355 – 359.

J. Han, et al. (2000b). 'Mining frequent patterns without candidate generation'. In *Proc. of ACM SIGMOD*, pp. 1–12.

S. Harms, et al. (2002). 'Discovering sequential association rules with constraints and time lags in multiple sequences'. In *International Symposium on Methodologies for Intelligent Systems (ISMIS)*, pp. 432–442.

R. Hilderman & H. Hamilton (1999). 'Knowledge discovery and interestingness measures: A survey'. Tech. Rep. 99-04, Department of Computer Science, University of Regina.

R. J. Hilderman & H. J. Hamilton (2001). 'Evaluation of Interestingness Measures for Ranking Discovered Knowledge'. *Lecture Notes in Computer Science* **2035**:247.

F. Hoeppner (2001). 'Discovery of Temporal Patterns - Learning Rules about the Qualitative Behaviour of Time Series'. In *Proc. of PKDD*, pp. 192–203.

F. Hoeppner & F. Klawonn (2001). 'Finding Informative Rules in Interval Sequences'. In *Advances in Intelligent Data Analysis, Proc. of the 4th International Symposium*, pp. 123–132.

S.-Y. Hwang, et al. (2004). 'Discovery of temporal patterns from process instances'. *Computers in Industry* **53**(3):345–364.

X. Ji, et al. (2007). 'Mining minimal distinguishing subsequence patterns with gap constraints'. *Knowledge and Information Systems* **11**(3):259–286.

P. Kam & A. W. Fu (2000). 'Discovering Temporal Patterns for Interval-Based Events.'. In *DaWaK*, pp. 317–326.

M. Kamber & R. Shinghal (1996). 'Evaluating the interestingness of characteristic rules'. In *Proc. of ACM SIGKDD*, pp. 263–266.

S. Laxman, et al. (2007). 'Discovering Frequent Generalized Episodes When Events Persist for Different Durations'. *IEEE Transactions on Knowledge and Data Engineering* **19**(9):1188–1201.

M. Leleu, et al. (2003). 'GO-SPADE: Mining Sequential Patterns over Databases with Consecutive Repetitions'. In *Proc. of MLDM*, pp. 293–306.

C. K.-S. Leung, et al. (2007). 'CanTree: a canonical-order tree for incremental frequent-pattern mining'. *Knowledge and Information Systems* **11**(3):287–311.

S. K. Liddell (1980). 'American Sign Language Syntax'. *The Hague: Mouton* .

J.-L. Lin (2002). 'Mining Maximal Frequent Intervals'. Tech. rep., Department of Information Management, Yuan Ze University.

J.-L. Lin (2003). 'Mining Maximal Frequent Intervals'. In *Proc. of SAC*, pp. 624–629.

H. Lu, et al. (1998). 'Stock Movement Prediction and n-dimensional Inter-Transaction Association Rules'. In *Proc. of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pp. 12:1–12:7.

C. Luo & S. M. Chung (2008). 'A scalable algorithm for mining maximal frequent sequences using a sample'. *Knowledge and Information Systems* **15**(2):149–179.

H. Mannila & H. Toivonen (1996). 'Discovering generalized episodes using minimal occurences'. In *Proc. of ACM SIGKDD*, pp. 146–151.

H. Mannila, et al. (1995). 'Discovering frequent episodes in sequences'. In *Proc. of ACM SIGKDD*, pp. 210–215.

F. Moerchen (2006). 'Algorithms For Time Series Knowledge Mining'. In *Proc. of ACM SIGKDD*.

C. Mooney & J. F. Roddick (2004). 'Mining Relationships Between Interacting Episodes'. In *Proc. of SDM*.

C. Neidle (2002a). 'SignStream: A Database Tool for Research on Visual-Gestural Language'. *Journal of Sign Language and Linguistics* **4**:203–214.

C. Neidle (2002b). 'SignStream Annotation: Conventions used for the American Sign Language

Linguistic Research Project'. *American Sign Language Linguistic Research Project Report* **11**.

C. Neidle (2003). 'Language Across Modalities: ASL Focus and Question Constructions'. *Linguistic Variation Yearbook* **2**:71–93.

C. Neidle, et al. (2000). 'The Syntax of American Sign Language: Functional Categories and Hierarchical Structure' .

C. Neidle & R. G. Lee (2006). 'Syntactic agreement across language modalities'. *Studies on Agreement* .

C. Neidle, et al. (2001). 'SignStream: A Tool for Linguistic and Computer Vision Research on Visual-Gestural Language Data'. *Behavior Research Methods, Instruments, and Computers* **33**:311–320.

B. Oezden, et al. (1998). 'Cyclic Association Rules'. In *Proc. of IEEE ICDE*, pp. 412–421.

E. R. Omiecinski (2003). 'Alternative Interest Measures for Mining Associations in Databases'. *IEEE Transactions On Knowledge and Data Engineering* **15**(1):39–79.

P. Papapetrou, et al. (2006). 'Discovering Frequent Poly-Regions in DNA Sequences'. In *Proc. of the IEEE ICDM Workshop on Data Mining in Bioinformatics.*

P. Papapetrou, et al. (2005). 'Discovering Frequent Arrangements of Temporal Intervals'. In *Proc. of IEEE ICDM*, pp. 354–361.

N. Pasquier, et al. (1999). 'Discovering frequent closed itemsets for association rules'. In *Proc. of ICDT*, pp. 398–416.

J. Pei, et al. (2000). 'CLOSET: An efficient algorithm for mining frequent closed itemsets'. In *Proc. of DMKD*, pp. 11–20.

J. Pei, et al. (2001). 'PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth'. In *Proc. of IEEE ICDE*, pp. 215–224.

J. Pei, et al. (2002). 'Constraint-based sequential pattern mining in large databases'. In *Proc. of CIKM*, pp. 18–25.

G. Piatetsky-Shapiro (1991). 'Discovery, Analysis and Presentation of Strong Rules'. *Knowledge Discovery in Databases* pp. 229–248.

M. Seno & G. Karypis (2002). 'SLPMiner: An algorithm for finding frequent sequential patterns using length-decreasing support constraint'. In *Proc. of IEEE ICDM*, pp. 418–425.

R. Srikant & R. Agrawal (1996). 'Mining Sequential Patterns: Generalizations and Performance Improvements'. In *Proc. of EDBT*, pp. 3–17.

Steinbach, et al. (2007). 'Generalizing the notion of confidence'. *Knowledge and Information Systems* **12**(3):279–299.

P. Tan & V. Kumar (2000). 'Interestingness Measures for Association Patterns: A Perspective'. Tech. Rep. TR00-036, Department of Computer Science, University of Minnesota.

P. Tan, et al. (2002). 'Selecting the right interestingness measure for association patterns'. In *Proc. of ACM SIGKDD*, pp. 183–192.

I. Tsoukatos & D. Gunopulos (2001). 'Efficient Mining of Spatiotemporal Patterns'. In *Proc. of the SSTD*, pp. 425–442.

R. Villafane, et al. (2000). 'Knowledge Discovery from Series of Interval Events'. *Intelligent Information Systems* **15**(1):71–89.

J. Wang & J. Han (2004). 'BIDE: Efficient Mining of Frequent Closed Sequences'. In *Proc. of IEEE ICDE*, pp. 79–90.

G. I. Webb (2006). 'Discovering Significant Rules'. In *Proc. of ACM SIGKDD.*

G. I. Webb & S. Zhang (2005). 'k-Optimal-Rule-Discovery'. *Data Mining and Knowledge Discovery* **10**:39–79.

E. Winarko & J. F. Roddick (2007). 'ARMADA - An algorithm for discovering richer relative temporal association rules from interval-based data'. *Data Knowl. Eng.* **63**(1):76–90.

S.-Y. Wu & Y.-L. Chen (2007). 'Mining Nonambiguous Temporal Patterns for Interval-Based Events'. *IEEE Transactions on Knowledge and Data Engineering* **19**(6):742–758.

D. Xin, et al. (2006). 'Discovering Interesting Patterns Through User's Interactive Feedback'. In *Proc. of ACM SIGKDD.*

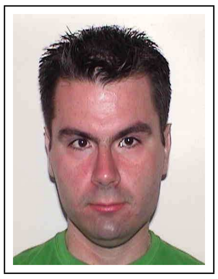
X. Yan, et al. (2003). 'CloSpan: Mining Closed Sequential Patterns in Large Databases'. In *Proc. of SDM.*

M. Zaki (2001). 'SPADE: An efficient algorithm for mining frequent sequences'. *Machine Learning* **40**:31–60.

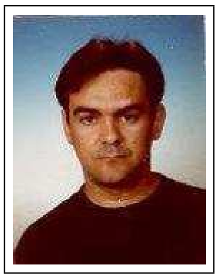
M. Zaki & C. Hsiao (2002). 'CHARM: An efficient algorithm for closed itemset mining'. In *Proc. of SIAM*, pp. 457–473.

M. J. Zaki (2000). 'Sequence Mining in Categorical Domains: Incorporating Constraints'. In *CIKM*, pp. 422–429.

Author Biographies



Panagiotis Papapetrou received the BSc degree in Computer Science from the University of Ioannina, Greece in 2003 and the MA degree in Computer Science from Boston University in 2006. Currently he is a PHD student in the Computer Science Department at Boston University. His research interests include data mining, knowledge discovery in databases, sequential and temporal pattern mining, gesture and sign language recognition, human motion analysis, efficient similarity-based retrieval and bioinformatics. He is a member of the IEEE Computer Society.



George Kollios received his Diploma in Electrical and Computer Engineering in 1995 from the National Technical University of Athens, Greece; and the M.Sc. and Ph.D. degrees in Computer Science from Polytechnic University, New York in 1998 and 2000 respectively. He is currently an Associate Professor in the Computer Science Department at Boston University in Boston, Massachusetts. His research interests include spatio-temporal databases and data mining, database security, multimedia indexing, and approximation algorithms in data management. He is the recipient of an NSF CAREER Award and his research is supported by NSF.He is currently an Associate Editor for the ACM Transactions on Database Systems and IEEE Transactions on Knowledge and Data Engineering. He has served in many technical program committees for top database and data mining conferences. He is a member of ACM and IEEE Computer Society.

Stan Sclaroff is a professor of Computer Science and the chair of the Department of Computer Science at Boston University. He received the PhD degree from the Massachusetts Institute of Technology in 1995. He founded the Image and Video Computing research group at Boston University in 1995. In 1996, he received a US Office of Naval Research (ONR) Young Investigator Award and a US National Science Foundation (NSF) Faculty Early Career Development Award. Since then, he has coauthored numerous scholarly publications in the areas of tracking, video-based analysis of human motion and gesture, deformable shape matching and recognition, as well as image/video database indexing, retrieval, and data mining methods. He has served on the technical program committees of more than 80 computer vision conferences and workshops. He has served as an associate editor for the IEEE Transactions on Pattern Analysis, 2000-2004 and 2006-present. He is a senior member of the IEEE.

Dimitrios Gunopulos received the PhD degree from Princeton University. He has held regular and visiting positions at the Max-Planck-Institute for Informatics, the University of Helsinki, the IBM Almaden Research Center, the Department of Computer Science and Engineering at the University of California, Riverside, and the Department of Informatics and Telecommunications at the University of Athens. His research is in the areas of data mining and knowledge discovery in databases, databases, sensor networks, peer-to-peer systems, and algorithms. He has coauthored more than 100 journal and conference papers that have been widely cited. He has served as a Program Committee co-Chair for IEEE ICDM 2008, ACM SIGKDD 2006, SSDBM 2003, and DMKD 2000, and he is currently an associate editor for the IEEE Transactions on Knowledge and Data Engineering, the IEEE Transactions on Parallel and Distributed Systems, and the ACM Transactions on Knowledge Discovery from Data. He is a member of the IEEE.

Correspondence and offprint requests to: Panagiotis Papapetrou, Department of Computer Science, Boston University, Boston, MA 02215, USA. Email: panagpap@cs.bu.edu