Eliciting Multi-Dimensional Relational Patterns

Nicola Di Mauro, Teresa M.A. Basile, Grazia Bombini, Stefano Ferilli, and Floriana Esposito

Università degli Studi di Bari, Dipartimento di Informatica, 70125 Bari, Italy
{ndm, basile, gbombini, ferilli, esposito}@di.uniba.it

Abstract. Here the issue of discovery of frequent multi-dimensional patterns from relational sequences is addressed. The great variety of applications of sequential pattern mining makes this problem one of the central topics in data mining. Nevertheless, sequential information may concern data on multiple dimensions and, hence, the mining of sequential patterns from multi-dimensional information results very important. This work takes into account the possibility to mine complex patterns, expressed in a first-order language, in which events may occur along different dimensions. Specifically, multi-dimensional patterns are defined as a set of atomic first-order formulae in which events are explicitly represented by a variable and the relations between events are represented by a set of dimensional predicates. A complete framework and an *Relational Learning* algorithm to tackle this problem are presented along with some experiments on artificial and real multi-dimensional sequences.

1 Introduction

The rapid growth of the amount of data stored in large databases has lead to an increasing interest in the data mining research area and, in particular, towards methods to discovery hidden structured patterns in large databases. The sequences are the simplest form of structured patterns and different methodologies have been proposed to face the problem of sequential pattern mining, firstly introduced by R. Agrawal and R. Srikant in [2], with the aim of capturing the existent maximal frequent sequences in a given database. The issue of discovering all frequent sequences of itemsets in a dataset is crucial when the data to be mined have some sequential nature like events in the case of temporal information. Furthermore, some real world domains such as user profiling, medicine, local weather forecast and bioinformatics show an inherent propension to be modelled by means of sequences of events/objects related to each other. This great variety of applications of sequential pattern mining makes this problem one of the central topics in data mining as showed by the research efforts produced in recent years [1, 23, 7, 18, 19] .

However, some environments involve very complex components and features. Thus, the classical existing data mining approaches, that look for patterns in a single data table, have been extended to the multi-relational data mining approaches that look for patterns involving multiple tables (relations) from a relational database. This has led to the exploitation of a more powerful knowledge representation formalism as first-order logic. Some works facing the problem of knowledge discovery from spatial and temporal data in multi-relational data mining research area are present in literature [16, 20, 5, 21, 14]. Nevertheless, there exists no contributions presenting a framework to manage the general case of multi relational data in which, for example, spatial and temporal information may co-exist.

On the other hand, it is worth to note that sequential information might concern data on multiple dimensions and, hence, the mining of sequential patterns from multi-dimensional information turns out to be very important. An attempt to propose a (two-dimensional) knowledge representation formalism to represent spatio-temporal information based on multi-dimensional modal logics is proposed in [3], while the first work presenting algorithms to mine multi-dimensional patterns has been presented in 2001 by Pinto et al. [19]. However, all the works in multi-dimensional data mining have been restricted to the propositional case, not involving a first-order representation formalism.

In this paper we provide an Inductive Logic Programming (ILP) [17] algorithm for discovering *first-order* (DATALOG) *maximal frequent patterns* in *multi-dimensional* relational sequences. Multi-dimensional patterns are defined as a set of atomic first-order formulae in which events are explicitly represented by a variable and the relations between events are represented by a set of dimensional predicates.

2 Mining Multi-Dimensional Patterns

We used Datalog [22] as representation language for the domain knowledge and patterns, that here is briefly reviewed. A *first-order alphabet* consists of variables, predicate symbols and function symbols (including constants). A *term* is defined as a constant symbol (a function symbol of arity 0, i.e. followed by a 0-tuple of terms) or a variable. An atom $p(t_1, \ldots, t_n)$ (or atomic formula) is a predicate symbol p of arity n applied to n terms t_i. Both l and its negation $\overline{l}$ are said to be *literals* (resp. positive and negative literal) whenever l is an atomic formula.

A *clause* is a formula of the form $\forall X_1 \forall X_2 \ldots \forall X_n (L_1 \vee L_2 \vee \ldots \vee \overline{L}_i \vee \overline{L}_{i+1} \vee \ldots \vee \overline{L}_m)$ where each L_i is a literal and $X_1, X_2, \ldots X_n$ are all the variables occurring in $L_1 \vee L_2 \vee \ldots \overline{L}_i \vee \ldots \overline{L}_m$. Most commonly the same clause is written as $L_1, L_2, \ldots \leftarrow L_i, L_{i+1}, \ldots L_m$. Clauses, literals and terms are said to be *ground* whenever they do not contain variables. A *Horn clause* is a clause which contains at most one positive literal. A *Datalog clause* is a clause with no function symbols of non-zero arity; only variables and constants can be used as predicate arguments.

A *substitution* θ is defined as a set of bindings $\{X_1 \leftarrow a_1, \ldots, X_n \leftarrow a_n\}$ where $X_i, 1 \leq i \leq n$ is a variable and $a_i, 1 \leq i \leq n$ is a term. A substitution θ is applicable to an expression e, obtaining the expression $e\theta$, by replacing all variables X_i with their corresponding terms a_i.

Definition 1 (1-dimensional relational sequence). *A 1-dimensional relational sequence may be defined as an ordered list of Datalog atoms separated by the operator $<$, $l_1 < l_2 < \cdots < l_n$.*

However, in order to make the proposed framework more general, the concept of *fluents* introduced by J. McCarthy in [15] should be considered: "After having defined a *situation*, s_t, as the complete state of the universe at an instant of time t, then a fluent is defined as a function whose domain is the space of situations. In particular, a *propositional fluent* ranges in (true,false). For example, raining(x, s_t) is true if and only if it is raining at the place x in the situation s_t."

If we consider a sequence as an ordered succession of events for each dimension, a fluent may be used to indicate that an atom is true for a given event. In particular, in our description language we can distinguish two kinds of Datalog atoms: *dimensional* and *non-dimensional* atoms. Specifically:

- non-dimensional atoms, that may be divided into
  - *fluent atoms*: explicitly referring to a given event (i.e., in which one of its argument denotes an event);
  - *non-fluent atoms*: denoting relations between objects (with arity greater than 1), or characterizing an object (with arity 1) involved in the sequence;
- dimensional atoms: referring to dimensional relations between events involved in the sequence.

The choice to add the event as an argument of the predicates is necessary for the general case of n-dimensional sequences with $n > 1$. In this case, indeed, the operator $<$ is not sufficient to express multi-dimensional relations and we must use its general version $<_i, 1 \leq i \leq n$. Specifically, $(e_1 <_i e_2)$ denotes that the event e_1 gives rise to the event e_2 in the dimension i. Hence, in our framework a multi-dimensional data is supposed to be a set of events, and to each dimension corresponds a sequence of events.

Definition 2 (Multi-dimensional relational sequence). *A multi-dimensional relational sequence is a set of Datalog atoms, involving k events and concerning n dimensions, in which there are non-dimensional atoms (fluents and non-fluents) and each event may be related to another event by means of the $<_i$ operators, $1 \leq i \leq n$.*

In order to represent multi-dimensional relational patterns, some dimensional operators must be introduced. The following symbols for describing general event relationships along many dimensions have been adopted. In particular, given a set $\mathcal{D}$ of dimensions, in the following are reported the multi-dimensional operators:

- $<_i$: *next step on dimension $i, \forall i \in \mathcal{D}$.* This operator indicates the direct successor on the dimension i. For instance, $(x <_{time} y)$ denotes that the event y is the direct successor of the event x on the dimension *time*. `next_i/2` is the corresponding Datalog predicate used to denote the successor operator;

– $\lhd_i$: *after some steps on dimension* $i, \forall i \in \mathcal{D}$. This operator encodes the transitive closure of $<_i$. For example, $(y \lhd_{spatialx} z)$ states that the event z occurs somewhere after the event y on the dimension *spatialx*. `follow_i/2` is the corresponding Datalog representation;
– $\bigcirc_i^n$: *exactly after* n *steps on dimension* $i, \forall i \in \mathcal{D}$. In particular it calculates the n-th direct successor. For instance, $(x \bigcirc_{spatialz}^n w)$ states that the event w is the n-th direct successor of the event x on the dimension *spatialz*. The `followat_1/3` Datalog predicate is used to represent such a situation.

Note that, the dimensional characteristics in the sequences will be described by using the $<_i$ operator, while the two dimensional operators $\lhd_i$ and $\bigcirc_i^n$, will be used, in combination with $<_i$ operator, to represent the frequent discovered patterns.

Definition 3 (Subsequence [9]). *Given a sequence* $\sigma = (e_1 e_2 \cdots e_m)$ *of* m *elements, a sequence* $\sigma' = (e'_1 e'_2 \cdots e'_k)$ *of length* k *is a subsequence (or pattern) of the sequence* σ *if*

1. $1 \leq k \leq m$
2. $\forall i, 1 \leq i \leq k, \exists j, 1 \leq j \leq m : e'_i = e_j$
3. $\forall i, j, 1 \leq i < j \leq k, \exists h, l, 1 \leq h < l \leq m : e'_i = e_h$ *and* $e'_j = e_l$.

The frequency of a subsequence in a sequence is the number of different mappings from elements of σ' *into the elements of* σ *such that the previous conditions hold.*

Note that this is a general definition of subsequence, in our case the *gaps* represented by the third condition are modelled by the $\lhd_i$ and $\bigcirc_i^n$ operators as reported in the following definition.

Definition 4 (Multi-dimensional relational pattern). *A multi-dimensional relational pattern is a set of Datalog atoms, involving* k *events and regarding* n *dimensions, in which there are non-dimensional atoms and each event may be related to another event by means of the operators* $<_i$, $\lhd_i$ *and* $\bigcirc_i^n$ *operators,* $1 \leq i \leq n$.

We are interested in finding maximal frequent patterns with a high frequency in long sequences. A pattern σ' of a sequence σ is *maximal* if there is no pattern σ'' of σ more frequent than σ' and such that σ' is a subsequence of σ''. In order to calculate the frequency of a pattern over a sequence it is important to define the concept of sequence subsumption.

If we indicate the operator $<_i$ with the Datalog predicate `next_i(X,Y)`, the Datalog definition of the operators $\bigcirc_i^k$ and $\lhd_i$ can be formulated as follows:

```
followat_i(1,X,Y) ← next_i(X,Y), !.
followat_i(K,X,Y) ← next_i(X,Z), K1 is K - 1, followat_i(K1,Z,Y).

follow_i(X,Y) ← next_i(X,Y).
follow_i(X,Y) ← next_i(X,Z), follow_i(Z,Y).
```

These definitions are added to the background knowledge $\mathcal{B}$ and used to prove the dimensional operators appearing in the patterns using the following definition of subsumption. Given S a multi-dimensional relational sequence, in the following we will indicate by Σ the set of Datalog clauses $\mathcal{B} \cup U$, where U is the set of ground atoms in S.

Definition 5 (Subsumption). *Given P a multi-dimensional relational pattern and S a multi-dimensional relational sequence, let $\Sigma = \mathcal{B} \cup U$. The pattern P* subsumes *the sequence S, written as $P \subseteq S$, iff there exists an SLD_{OI}-deduction of P from Σ.*

An $\mathrm{SLD}_{\mathrm{OI}}$-deduction is an SLD-deduction under Object Identity. In the Object Identity framework, within a clause, terms that are denoted with different symbols must be distinct, i.e. they must represent different objects of the domain.

3 The algorithm

After having defined the formalism for representing sequences and patterns, here we describe the algorithm for frequent multi-dimensional relational pattern mining based on the same idea of the generic level-wise search method, known in data mining from the Apriori algorithm [1]. The level-wise algorithm makes a breadth-first search in the lattice of patterns ordered by a specialization relation $\preceq$. The search starts from the most general patterns, and at each level of the lattice the algorithm generates candidates by using the lattice structure and then evaluates the frequencies of the candidates. In the generation phase, some patterns are taken out using the monotonicity of pattern frequency (if a pattern is not frequent then none of its specializations is frequent).

The mining method is outlined in Algorithm 1. The generation of the frequent patterns is based on a top-down approach. The algorithm starts with the most general patterns. These initial patterns are all of length 1 and are generated by adding to the empty pattern a non-dimensional atom. Successively, at each step it tries to specialize all the potential frequent patterns, discarding the non-frequent patterns and storing the ones whose length is equal to the user specified input parameter *maxsize*. Furthermore, for each new refined pattern, semantically equivalent patterns are detected, by using the θ_{OI}-subsumption relation, and discarded. Note that the length of a pattern is defined as the number of non-dimensional atoms. In the specialization phase, the specialization operator under θOI-subsumption is used. Basically, the operator adds atoms to the pattern.

The algorithm uses a background knowledge $\mathcal{B}$ (a set of Datalog clauses) containing the sequence and a set of constraints that must be satisfied by the generated patterns. In particular $\mathcal{B}$ contains:

– *maxsize(M)*: maximal pattern length (i.e., the maximum number of non-dimensional predicates that may appear in the pattern);

Algorithm 1 MDLS

```
Require: $\Sigma = \mathcal{B} \cup U$, where $\mathcal{B}$ is the background knowledge and $U$ is the set of ground atoms in the sequence $S$.
Ensure: $P_{max}$: the set of maximal frequent patterns
 1: $P \leftarrow$ { initial patterns }
 2: $P_{max} \leftarrow \emptyset$
 3: while $P \neq \emptyset$ do
 4:   $P_s \leftarrow \emptyset$
 5:   for all $p \in P$ do
 6:     /* generation step */
 7:     $P_s \leftarrow P_s \cup$ {all the specializations of $p$ that satisfy all the constraints posconstraints, negconstraints or atmostone}
 8:   $P \leftarrow \emptyset$
 9:   for all $p \in P_s$ do
10:     /* evaluation step */
11:     if freq($p$) $\geq$ minfreq then
12:       if length($p$) = maxsize then
13:         $P_{max} \leftarrow P_{max} \cup \{p\}$
14:       else
15:         $P \leftarrow P \cup \{p\}$
```

- *minfreq(m)*: this constraint indicates that the frequency of the patterns must be larger that m;
- *dimension(next_i)*: this kind of atom indicates that the sequence contains events on the dimension i. One can have more that one of such atoms, each of which denoting a different dimension. In particular, the number of these atoms represents the number of the dimensions.
- *type(p)*: denotes the type of the predicate's arguments p;
- *mode(p)*: denotes the input output mode of the predicate's arguments p;
- *negconstraint([$p_1, p_2, \ldots, p_n$])*: specifies a constraint that the patterns must not fulfill, i.e. if the clause $(p_1, p_2, \ldots, p_n)$ subsumes the pattern then it must be discarded. For instance, negconstraint([p(X,Y),q(Y)]) discards all the patterns subsumed by the clause (p(X,Y),q(Y));
- *posconstraint([$p_1, p_2, \ldots, p_n$])*: specifies a constraint that the patterns must fulfill. It discards all the patterns that are not subsumed by the clause $(p_1, p_2, \ldots, p_n)$;
- *atmostone([$p_1, p_2, \ldots, p_n$])*: this constraint discards all the patterns that make true more than one predicate among $p_1, p_2, \ldots, p_n$. For instance, atmostone([red(X),blue(X),green(X)]) indicates that each constant in the pattern can assume at most one of red, blue or green value;
- *key([$p_1, p_2, \ldots, p_n$])*: it is optional and specifies that each pattern must have one of the non-dimensional predicates $p_1, p_2, \ldots p_n$ as a starting literal.

The use of the `dimension(next_i)` literals, that specify the number of dimensions the sequence is based on, allows to the corresponding definitions of the predicates `followat_i/3` and `follow_i/2` to be automatically generated and added to the background knowledge $\mathcal{B}$.

Classical mode and type declarations are used to specify a language bias indicating which predicates can be used in the patterns and to formulate constraints on the binding of variables. The solution space is further pruned by using some positive and negative constraints specified by the `negconstraint` and `posconstraint` literals. The last pruning choice is defined by the `atmostone` literals. This last constraint is able to describe that some predicates are of the same type.

Since each pattern a) must start with a non-dimensional predicate, or with a predefined key, and b) its frequencey must be less than the sequence length, the frequency of a pattern can be defined as follows.

Definition 6 (Frequency). *Given a multi-dimensional relational pattern $P = (p_1, p_2, \ldots, p_n)$ and S a multi-dimensional relational sequence, the* frequency *of pattern P is equal to the number of different ground literals used in all the possible SLD_{OI}-deductions of P from $\Sigma = \mathcal{B} \cup U$ that make true the literal p_1.*

The refinement of pattern is obtained by using a refinement operator ρ that maps each pattern to a set of specializations of the pattern, i.e. $\rho(p) \subset \{p' | p \preceq p'\}$ where $p \preceq p'$ means that p is more general of p' or that p subsumes p'. In particular, given the set $\mathcal{D}$ of dimensions, the set $\mathcal{F}$ of fluent atoms, the set $\mathcal{P}$ of non-fluent atoms, the refinement operator for specializing the patterns is defined as follows:

adding a non-dimensional atom
- the pattern S is specialized by adding a non-dimensional atom $F \in \mathcal{F}$ (a fluent) referring to an event already introduced in S;
- the pattern S is specialized by adding a non-dimensional atom $P \in \mathcal{P}$;

adding a dimensional atom
- the pattern S is specialized by adding the dimensional atom $(x <_i y)$ $i \in \mathcal{D}$, relating the events x and y, iff $\exists$ a fluent $F \in \mathcal{F}$ in S which event argument is x and there not exist the atoms $(x \lhd_i y)$ and $(x \bigcirc_i^n y)$ in S;
- the pattern S is specialized by adding the dimensional atom $(x \lhd_i y)$ $i \in \mathcal{D}$, relating the events x and y, iff $\exists$ a fluent $F \in \mathcal{F}$ in S which event argument is x and there not exist the atoms $(x <_i y)$ and $(x \bigcirc_i^n y)$ in S;
- the pattern S is specialized by adding the dimensional atom $(x \bigcirc_i^n y)$ $i \in \mathcal{D}$, relating the events x and y, iff $\exists$ a fluent $F \in \mathcal{F}$ in S which event argument is x and there not exist the atoms $(x <_i y)$ and $(x \lhd_i y)$ in S.

The dimensional atoms are added iff there exists a fluent atom referring to its starting event. This is to avoid unuseful chains of dimensional predicates like this `p(`e_1`,a) (`$e_1 <_i e_2$`) (`$e_2 <_i e_3$`) (`$e_3 <_i e_4$`)`, that is naturally subsumed by `p(`e_1`,a) (`$e_1 \bigcirc_i^3 e_4$`)`. We recall that the length of a pattern P is equal to the number of non-dimensional atoms in P.

4 Experiments

MDSL has been implemented in Yap Prolog and evaluated by making some experiments on an artificial dataset and on trace files collected from different users of Unix csh [6, 9]. The analysis of the use of Unix command shell represents one of the classic applications in the domain of adaptive user interfaces and user modeling. Greenberg [6] collected logs from 168 users of the unix csh, divided into 4 target groups: 55 novice programmers, 36 experienced programmers, 52 computer scientists and 25 non-programmers. Table 1 reports statistics of finding frequent patterns for 3 users logs from the Greenberg dataset.

Each Greenberg's log file corresponding to a user is divided into login sessions denoted by a starting and an ending time record. Each command entered in each session has been annotated with the current working directory, alias substitution, history use and error status. Furthermore, each command name may be followed by some otpions and some parameters. For instance the command `ls -a *.c` has name `ls`, option `-a` and parameter `*.c`.

As pointed out in [9], this problem is a relational problem, since commands are interrelated by their execution order (or time), and each command can be eventually related to one or more parameters. A shell log may be viewed as a 2-dimensional sequence, since each command is followed by another command (the first dimension) and each command line is composed by an ordered sequence of tokens (i.e., command name, options and parameters). Each shell log has been represented as a set of logical ground atoms as follows.

- `command(e)` is the predicate used to indicate that `e` is a command. The command name has been used as a predicate symbol applied to `e`;
- `parameter(e,p)` has been used to indicate that `e` has the parameter `p`. The parameter name has been used as a predicate symbol applied to `p`;
- `current_directory(c,d)` indicates that `d` is the current directory of the command `c`;
- `next_c(c1,c2)` ($<_c$) indicates that the command `c2` is the direct command successor of `c1`;
- `next_p(p1,p2)` ($<_p$) indicates that the parameter `p2` is the direct parameter successor of `p1`.

For instance the following shell log

```
cp paper.tex newpaper.tex
latex newpaper
xdvi newpaper
```

should be translated as

```
command(c1), '$cp'(c1),
   next_p(c1,c1p1), parameter(c1p1,'paper.tex'),
   next_p(c1p1,c1p2), parameter(c1p2,'newpaper.tex'),
next_c(c1,c2), '$latex'(c2),
   next_p(c2,c2p1), parameter(c2p1,'newpaper'),
next_c(c2,c3), '$xdvi'(c3),
   next_p(c3,c3p1), parameter(c3p1,'newpaper')
```

In this way it is possible to describe patterns such as

```
command(L), '$latex'(L),
   next_p(L,LP), parameter(LP,P),
next_c(L,X), '$xdvi'(X),
   next_p(X,XP), parameter(XP,P)
```

Table 1 reports statistics on MDLS performance on the Greenberg dataset. The first four columns denote, respectively, the user name, the number of literals of the sequence, the number of sessions for each user log file and the total number of commands in the log file. For each user some experiments has been made. The kind of experiment carried out is denoted in the fifth column that reports the operators used in the experiment. For instance $<_C$ and $<_P$ indicate that only these two operators have been used in the experiment. We see that, as the number of commands and dimensional operators grows, the execution time increases. Note that each session represents a separate sequence and a log file is a collection of sequences. There is no correlation between two session in a log file.

Table 1. MDLS performances (time in secs.). |S|: n. of literals in the sequence; |Ses|: n. of sessions for user log file; |C|: total number of commands in the log file; Op: dimensional operators used; L: max length of the patterns; F: min freq of the patterns; |MP|: n. of found maximal patterns; Sp: required specializations.

User	|S|	|Ses|	|C|	Op	L	F	Time	|MP|	Sp
				$<_C$	5	5	1.17	45	3597
				$<_C <_P$	5	5	2.68	45	9513
n9	2654	73	357	$<_C \bigcirc_C^n$	5	5	2.58	91	8495
				$<_C \lhd_C \bigcirc_C^n$	5	5	28.94	149	29081
				$<_{C,P} \lhd_{C,P} \bigcirc_{C,P}^n$	5	5	99.06	218	76299
				$<_C$	5	10	2.36	38	4512
				$<_C <_P$	5	10	3.58	18	6434
n17	5366	61	848	$<_C \bigcirc_C^n$	5	10	7.95	47	10808
				$<_C \lhd_C \bigcirc_C^n$	5	10	37.06	39	19198
				$<_{C,P} \lhd_{C,P} \bigcirc_{C,P}^n$	5	10	144.38	25	25142
				$<_C$	5	15	6.10	64	7716
				$<_C <_P$	5	15	19.33	77	24046
				$<_C \bigcirc_C^n$	5	15	16.30	139	20744
				$<_C \lhd_C \bigcirc_C^n$	5	15	138.06	162	51363
n7	12355	80	1231	$<_C$	5	80	1.78	9	953
				$<_C <_P$	5	80	4.06	11	2493
				$<_C \bigcirc_C^n$	5	80	4.14	16	2479
				$<_C \lhd_C \bigcirc_C^n$	5	80	42.55	29	6745
				$<_{C,P} \lhd_{C,P} \bigcirc_{C,P}^n$	5	80	164.99	49	17505

5 Related Work

As already pointed out, the problem of sequential pattern mining is a central one in a lot of data mining applications and many efforts have been done in order to propose purposely designed methods to face it. Most of the works have been restricted to propositional patterns, that is, patterns not involving first order predicates. One of the early domains that highlights the need to describe with structural information the sequences was the bioinformatic. Thus, the need to represent many real world domains with structured data sequences became more unceasing, and consequently many efforts have been done to extend existing or propose new methods to manage sequential patterns in which first order predicates are involved. On the other hand, for a fair description of some application domains, the sequences must involve not only relational objects but also the evolution of each object in more than one dimension. Unfortunately, to our knowledge, there are no methods able to manage sequences whose description involves both relations and more than one dimensions. In the following a brief survey of the techniques proposed to deal with relational or multi-dimensional sequences is presented along with the modelled application domain.

In [9] is presented a work, in the domain of user modelling, that helps shell users by creating scripts (a sequence of commands) from shell logs, that automate frequent performed tasks. The authors see this task as a relational learning problem, indeed commands may be interrelated by their execution order, and each command is possibly related to one or more parameters, giving out a representation of a shell log as a set of logical ground atoms. After having transformed shell logs in a relational representation, they applied the Warmr [5] system, an upgrade of the propositional Apriori algorithm that can detect first order logic association rules, for generating scripts. They used a specific predicate to specify that two commands are considered next to each other in a sequence.

Warmr [5] is based on the level-wise search of conventional association rule learning systems of the Apriori-family [1]. It extends these systems by looking for frequent patterns that may be expressed as conjunction of first-order literals. In Warmr a pattern is defined as a conjunction of first order literals. It performs a top-down level-wise search, starting with the key and refining patterns by adding literals to them. Infrequent patterns (i.e. patterns whose frequency is below a predefined threshold) are pruned as are their refinements. With Warmr it is possible to generate patterns that are syntactically different but semantically equivalent. This is due to the redundant conditions that may be added to a pattern or to the fact that the same pattern may be expressed in different ways.

As already described in [4], this problem may be avoided by using the Warmr's configurable language bias or by its constraint specification language. However, this solution does not solve the problem at all. Indeed, the constraints that may be defined in Warmr, by using its constraint specification language, are only syntax based, and they are not sufficient to handle semantic dependencies. For this and other limitations already described in [9, 8], in some cases the Warmr system is not able to calculate frequent subsequences and it is difficult to correctly represent the specific sequence mining task.

In [13] are presented a logic language, SeqLog, for mining sequences of logical atoms, and the inductive mining system MineSeqLog, that combines principles of the level-wise search algorithm with the version space in order to find all patterns that satisfy a constraint by using an optimal refinement operator for SeqLog. SeqLog is a logic representational framework that adopts two operators to represent the sequences: one to indicate that an atom is the direct successor of another and the other to say that an atom occurs somewhere after another. Furthermore, based on this language, the notion of subsumption, entailment and a fix point semantic are given. However, with SeqLog one can represent unidimensional sequences only.

In [12] it is proposed an extension of classical Fisher kernels, working on sequences over flat alphabets, in order to make they able to model logical sequences, i.e., sequences over an alphabet of logical atoms. Fisher kernels were developed to combine generative models with kernel methods, and have shown promising results for the combinations of support vector machines with (logical) hidden Markov models and Bayesian networks. Successively, in [10] the same authors proposed an algorithm for selecting logical hidden Markov models from data. Hidden Markov models are one of the most popular methods for analyzing sequential data, but they can be exploited to handle sequence of flat/unstructured symbols. The proposed logical extension [11] overcomes such weakness by handling sequences of structured symbols by means of a probabilistic ILP framework.

The work above reported extend/propose techniques to mine sequences involving relational objects. However, these methods, both logical and propositional, do not mention the possibility to manage patterns in which more than one dimension is take into account. On the other hand, it is not wrong to affirm that for most of the applications of real world domain generally the pattern sequences deal with different events each of which should be associated with different dimensions.

6 Conclusions

The issue of discovering sequential patterns from sequence data have drawn a lot of research efforts both in single data table and in multiple data table, known as multi-relational data. Although much work has been done in the area, no previous research revealed ways to find sequential patterns from multidimensional sequence data and in particular sequential patterns from multi-dimensional sequence expressed in first-order logic. Indeed, some works faces the problem of knowledge discovery from spatial and temporal data in the multi-relational data mining research area but there exists no contributions to manage the general case of multi-dimensional data in which, for example, spatial and temporal information may co-exist. Other works on multi-dimensional data mining, in some cases thinking to the concept of multi-dimension of a sequence as the presence of multiple attributes in data descriptions, have been restricted to the propositional case, not involving a first-order representation formalism. Finally, other works propose a (two-dimensional) knowledge representation formalism to represent spatio-temporal information based on multi-dimensional modal logics.

In this paper we proposed a logical framework for mining multi-dimensional patterns in which many dimensions can be specified. What we can obtain are maximal frequent multi-dimensional patterns described in a first order language. One of the most important characteristic of using logical framework for sequences is that we can incorporate additional information by using a background knowledge, and that any relation between atoms can be expressed or learned. The result is a dedicated system in which are incorporated specific language bias for multi-dimensional data in order to rise a faster execution and a smaller search space.

Acknowledgements

The authors would like to thank Jan Ramon for giving useful suggestions on setting the system Warmr and Saul Greenberg for providing the test Unix log data.

This work is partially funded on the Apulian Regional Project DDTA "Distretto Digitale a Supporto della Filiera Produttiva del Tessile-Abbigliamento".

References

1. R. Agrawal, H. Manilla, R. Srikant, H. Toivonen, and A.I. Verkamo. Fast discovery of association rules. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI Press, 1996.
2. R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings of the Int. Conf. on Data Engineering (ICDE95)*, pages 3–14, 1995.
3. Brandon Bennett, Anthony G. Cohn, Frank Wolter, and Michael Zakharyaschev. Multi-dimensional modal logic as a framework for spatio-temporal reasoning. *Applied Intelligence*, 17(3):239–251, 2002. Submitted 2000.
4. H. Blockeel, J. Fürnkranz, A. Prskawetz, and F. Billari. Detectin temporal change in event sequences: an application to demographic data. In *Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 29–41. Springer, 2001.
5. L. Dehaspe and H. Toivonen. Discovery of frequent datalog patterns. *Data Mining and Knowledge Discovery*, 3(1):7–36, 1999.
6. S. Greenberg. Using unix: collected traces of 168 users. Research Report 88/333/45, Department of Computer Science, University of Calgary, Alberta, 1988.
7. J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *ACM-SIGMOD Int. Conf.Management of Data (SIGMOD'00)*, pages 1–12, 2000.
8. N. Jacobs. *Relational Sequence Learning and User Modelling.* PhD thesis, Department of Computer Science, K.U.Leuven, Leuven, Belgium, October 2004.
9. N. Jacobs and H. Blockeel. From shell logs to shell scripts. In C. Rouveirol and M. Sebag, editors, *Proceedings of the 11th International Conference on Inductive Logic Programming*, volume 2157, pages 80–90. Springer, 2001.

10. K. Kersting, L. De Raedt, , and T. Raiko. Logical hidden markov models. *Journal of Artificial Intelligence Research - JAIR*, 25:425–456, April 2006. Submitted 2005.
11. K. Kersting and T. Raiko. 'Say EM' for selecting probabilistic models for logical sequences. In F. Bacchus and T. Jaakkola, editors, *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI05)*, pages 300–307. AUAI Press, 2005.
12. Kristian Kersting and Thomas Gärtner. Fisher kernels for logical sequences. In Jean-François Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino Pedreschi, editors, *Machine Learning: ECML 2004, Proceedings of the 15th European Conference on Machine Learning*, volume 3201 of *Lecture Notes in Computer Science*, pages 205–216. Springer, 2004.
13. S.D. Lee and L. De Raedt. Constraint based mining of first order sequences in SeqLog. In R. Meo, P.L. Lanzi, and M. Klemettinen, editors, *Database Support for Data Mining Applications*, volume 2682 of *LNCS*, pages 155–176. Springer, 2004.
14. D. Malerba and F.A. Lisi. Discovering associations between spatial objects: An ilp application. In *Proceedings of the 11th International Conference on Inductive Logic Programming*, volume 2157 of *LNCS*, pages 156–166. Springer, 2001.
15. J. McCarthy and P.J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969. reprinted in McC90.
16. S. Moyle and S. Muggleton. Learning programs in the event calculus. In *Proceedings of the 7th International Workshop on Inductive Logic Programming*, pages 205–212. Springer, 1997.
17. S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19/20:629–679, 1994.
18. P.J. Pei, J. Han, and W. Wang. Mining sequential patterns with constraints in large databases. In *Proceedings of the 11th ACM International Conference on Information and Knowledge Management*, pages 18–25, 2002.
19. H. Pinto, J. Han, J. Pei, K. Wang, Q. Chen, and U. Dayal. Multi-dimensional sequential pattern mining. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pages 81–88, New York, NY, USA, 2001. ACM Press.
20. L. Popelínsky. Knowledge discovery in spatial data by means of ILP. In *Proceedings of the Second European Symposium on Principles of Data Mining and Knowledge Discovery*, pages 185–193. Springer, 1998.
21. J. Rodríguez, C. Alonso, and H. Böstrom. Learning first order logic time series classifiers. In J. Cussens and A. Frisch, editors, *Proceedings of the 10th International Workshop on Inductive Logic Programming*, pages 260–275. Springer, 2000.
22. J.D. Ullman. *Principles of Database and Knowledge-Base Systems*, volume I. Computer Science Press, 1988.
23. M.J. Zaki. SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning Journal: Special issue on Unsupervised Learning*, 42(1/2):31–60, 2001.