Multi-Relational Pattern Mining Based-on Combination of Properties with Preserving Their Structure in Examples

Yusuke Nakano and Nobuhiro Inuzuka

Nagoya Institute of Technology,
Gokiso-cho, Showa, Nagoya 466-8555, Japan
nakano18115115@gmail.com, inuzuka@nitech.ac.jp
Phone: +81-52-735-5050, Facsimile: +81-52-735-5473

Abstract. We propose an algorithm for multi-relational pattern mining through the problem established in WARMR. In order to overcome the combinatorial problem of large pattern space, another algorithm MAPIX restricts patterns into combination of basic patterns, called properties. A property is defined as a set of literals appeared in examples and is of an extended attribute-value form. Advantage of MAPIX is to make patterns from pattern fragments occurred in examples. Many patterns which are not appeared in examples are not tested. Although the range of patterns is clear and MAPIX enumerates them efficiently, a large part of patterns are out of the range. The proposing algorithm keeps the advantage and extends the way of combination of properties. The algorithm combines properties as they appeared in examples, we call it structure preserving combination.

1 Introduction

This paper studies multi-relational pattern mining obeying the line of WARMR[2, 3]. WARMR generates and tests candidate patterns with a pruning technique similar to Apriori[1]. In spite of the cut-down procedure it has the exponentially growing space of hypothesis with respect to the length of patterns and the number of relations. Another algorithm MAPIX restricts patterns into conjunctions of basic patterns, called properties. A property is an extended attribute-value form consisting of literals that refer to parts of objects and a literal that describes a property of or a relation among the parts. For example for a person (or his/her family) a basic pattern consists of referential literals (or an extended attribute), for example "one of his/her grandchildren", and a descriptive literal (or an extended attribute value), for examples "it is male" and then it describes a basic property, for example "this person has a grandson".

MAPIX finds patterns made of properties appeared in samples relying on type and mode information of relations. The search can be seen as a combination of bottom-up and top-down search, it constructs properties from samples in a bottom-up way and tests patterns combining the properties in a top-down way.

The properties of MAPIX and their conjunctions are natural but the range in which MAPIX generates patterns is still narrow. There seems another natural range of patterns to be generated. We propose another way to combine

properties. For example, for a person of a family, "having a son" and "having a granddaughter" are properties and by combining them we may have "having a son and a granddaughter". But it may not represent a real situation of the family precisely, in which it may happen that "having a son who has a daughter".

We propose an algorithm by using a structural combination[4] of basic patterns. We propose to use preserved structure in a sample and give a simple algorithm. Our algorithm generates patterns in a larger pattern space.

2 Patterns and MAPIX Algorithm

Datalog is used to represent data and patterns. Datalog clauses are of the form $\forall(h \leftarrow b_1 \wedge \ldots \wedge b_n)$ without functors, where $\forall F$ means all variables in F are universally quantified and $\forall$ is omitted when understood. For $c = h \leftarrow b_1 \wedge \ldots \wedge b_n$, head($c$) denotes the head atom h and body(c) denotes the body conjunction $b_1 \wedge \ldots \wedge b_n$. A fact is a clause without body. A substitution is described by $\theta = \{v_1/t_1, \ldots, v_n/t_n\}$ for variables v_i and terms t_i. $P\theta$ for a formula P means replacing every variable v_i with t_i.

For our mining task a Datalog DB $\boldsymbol{R}$ is given and one of extensional relations is specified for a *target* (It corresponds to the concept *key* of WARMR). A fact of the target relation is called a *target instance*.

A *query* is a clause without head $\leftarrow b_1 \wedge \ldots \wedge b_n$, equivalently an existentially quantified conjunction $\exists(b_1 \wedge \ldots \wedge b_n)$, where $\exists Q$ means all variables in Q are existentially quantified. When a formula is clearly meant to be a query $\exists$ is dropped. A query q is said to succeed wrt $\boldsymbol{R}$ when $\boldsymbol{R} \models \exists q$.

The following gives patterns, among which we are interested in frequent ones.

Definition 1 (pattern). *A pattern is a Datalog clause whose head is of the target predicate. For a target instance e and a pattern P, $P(e)$ denotes a query $\exists(body(P)\theta)$ where θ is the mgu (most general unifier) of e and head(P). When $P(e)$ succeeds we say that e* possesses *P.*

Definition 2 (frequent pattern). *The* frequency *of P is the number of target instances which possess P. P is* frequent *if its frequency exceeds $sup_{min} \cdot N$, where sup_{min} is a given minimal support and N is the number of all target instances.*

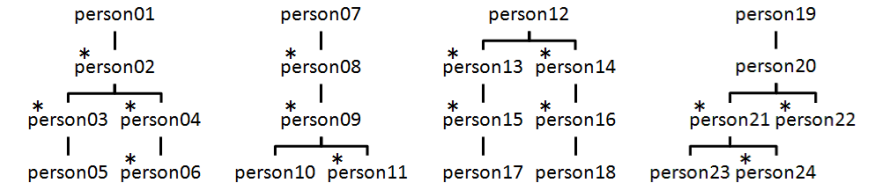
Example 1 (running example). Let us consider a DB $\boldsymbol{R}_{\mathbf{fam}}$ on families (Fig. 1). It includes four relations, $\mathsf{parent}(x, y)$ meaning x is a parent of y, $\mathsf{female}(x)$ for a female x, $\mathsf{male}(x)$ for a male x, and $\mathsf{grandfather}(x)$ meaning x is someone's grandfather. We use gf, p, m, f for the relations for short. We also abbreviate `person01` as `01`. Let gf be a target.

Then, for example the following formula is a pattern.

$$P = \mathsf{gf}(A) \leftarrow \mathsf{m}(A) \wedge \mathsf{p}(A, B) \wedge \mathsf{f}(B)$$

For a target instance $\epsilon = \mathsf{gf}(\mathtt{01})$, $P(\epsilon)$ denotes a query,

$$P(\epsilon) = \exists((\mathsf{m}(A) \wedge \mathsf{p}(A, B) \wedge \mathsf{f}(B))\theta) = \exists(\mathsf{m}(\mathtt{01}) \wedge \mathsf{p}(\mathtt{01}, B) \wedge \mathsf{f}(B)).$$



grandfather
person01
person07
person12
person19
person20

parent	
person01	person02
person02	person03
person02	person04
person03	person05
...	...

male
person01
person05
person07
person10
...

female
person02
person03
person04
person06
...

Fig. 1. The family DB $R_{\mathbf{fam}}$, including grandfather, parent, male and female, of which grandfatheris a target. The persons with $*$ are female and others are male.

where θ is the mgu of ϵ and head(P). The query $P(\epsilon)$ succeeds by an assignment $\{B \mapsto \mathsf{02}\}$ then e possesses P. □

Many ILP algorithms assume modes for predicates to restrict patterns. Some arguments of a predicate have a role as input (denoted by $+$) and some as output ($-$). We give parent$(+,-)$, male$(+)$, and female$(+)$ to the predicates in $R_{\mathbf{fam}}$.

We distinguish between two classes of predicates. Predicates with at least one $\langle - \rangle$-arg. are called *path predicates*, e.g. parent$(+,-)$, which have a role like a function generating a term from others. Predicates without $\langle - \rangle$-arg. are called *check predicates*, e.g. male$(+)$ and female$(+)$, which have a role describing a property of given terms. An instance of a path/check predicate in DB is called a path/check literal. We do not give mode for target predicate.

Using these concepts MAPIX extracts basic patterns, called *properties*, from DB and generalize them to basic patterns, called *property items*.

Definition 3 (property). *A* property *of a target instance e wrt DB $\boldsymbol{R}$ is a minimal set L of ground atoms in $\boldsymbol{R}$ satisfying*

1. *L includes exactly one check literal, and*
2. *L can be given a linear order where every term in a $\langle + \rangle$-arg. of a literal in L is occurred in some precedent literals in the order or e.*

Definition 4 (variablization). *For a ground formula α a formula β is a* variablization *of α when*

1. *β does not include any ground term, and*
2. *there exists a substitution $\theta = \{v_1/t_1, \cdots, v_n/t_n\}$ that satisfies (a) $\alpha = \beta\theta$ and (b) $t_1, \ldots, t_n$ in θ are all different terms in α.*

We assume to use new variables never used before when variablizing.

Definition 5. *For a set $L = \{l_1, \cdots, l_m\}$ of ground literals and a target instance e* $\mathbf{var}(e \leftarrow L)$ *denotes a variablization of $e \leftarrow l_1 \wedge \ldots \wedge l_m$.*

Table 1. Properties and property items of $\epsilon = \mathsf{gf}(01)$.

$\mathsf{pr0} = \{\mathsf{m}(01)\}$	$\mathsf{it0} = \mathsf{gf}(A) \leftarrow \mathsf{m}(A)$
$\mathsf{pr1} = \{\mathsf{p}(01,02),\mathsf{f}(02)\}$	$\mathsf{it1} = \mathsf{gf}(A) \leftarrow \mathsf{p}(A,B) \wedge \mathsf{f}(B)$
$\mathsf{pr2} = \{\mathsf{p}(01,02),\mathsf{p}(02,03),\mathsf{f}(03)\}$ ($\{\mathsf{p}(01,02),\mathsf{p}(02,04),\mathsf{f}(04)\}$)	$\mathsf{it2} = \mathsf{gf}(A) \leftarrow \mathsf{p}(A,B) \wedge \mathsf{p}(B,C) \wedge \mathsf{f}(C)$
$\mathsf{pr3} = \{\mathsf{p}(01,02),\mathsf{p}(02,03),\mathsf{p}(03,05),\mathsf{m}(05)\}$	$\mathsf{it3} = \mathsf{gf}(A) \leftarrow \mathsf{p}(A,B) \wedge \mathsf{p}(B,C) \wedge \mathsf{p}(B,D) \wedge \mathsf{m}(D)$
$\mathsf{pr4} = \{\mathsf{p}(01,02),\mathsf{p}(02,04),\mathsf{p}(04,06),\mathsf{f}(06)\}$	$\mathsf{it4} = \mathsf{gf}(A) \leftarrow \mathsf{p}(A,B) \wedge \mathsf{p}(B,C) \wedge \mathsf{p}(B,D) \wedge \mathsf{f}(D)$

When L is a property of e, $\mathbf{var}(e \leftarrow L)$ is called a *property item* of e. Possessing P by e and a query $P(e)$ are defined as in Definition 1.

Example 2. $L = \{\mathsf{p}(01,02),\mathsf{p}(02,03),\mathsf{f}(03)\}$ is a property of $\epsilon = \mathsf{gf}(01)$. Then $\mathsf{it} = \mathbf{var}(\epsilon \leftarrow L) = \mathsf{gf}(A) \leftarrow \mathsf{p}(A,B) \wedge \mathsf{p}(B,C) \wedge \mathsf{f}(C)$ is possessed by ϵ, i.e. $\boldsymbol{R}_{\mathbf{fam}} \models \mathsf{it}(\epsilon)$. □

Then, MAPIX algorithm is as follows:

1. It samples an appropriate number of target instances from a target relation.
2. For each sampled instance it extracts property items hold on DB.
3. It executes an Apriori-like level-wise frequent pattern mining algorithm by regarding the satisfaction of a property item as possession of it.

As discussed in [5] the size of sampled instances in step 1 can be constant with respect to the the size of all examples.

Example 3. Table 1 shows property items produced from $\mathsf{gf}(01)$. Only these are frequent for min sup 60% even if we sample all instances, when another infrequent property items $\mathsf{gf}(A) \leftarrow \mathsf{p}(A,B) \wedge \mathsf{p}(B,C) \wedge \mathsf{m}(C)$ is extracted from $\mathsf{gf}(20)$.

In step 3 MAPIX combines property items by a simple conjunction. For example $\mathsf{it2}$ (means having a granddaughter) and $\mathsf{it4}$ (means having a great-granddaughter) are combined to the following pattern, which we write as $\langle \mathsf{it2}, \mathsf{it4} \rangle$,

$$\langle \mathsf{it2}, \mathsf{it4} \rangle = \mathsf{gf}(A) \leftarrow \mathsf{p}(A,B) \wedge \mathsf{p}(B,C) \wedge \mathsf{f}(C) \wedge \mathsf{p}(A,D) \wedge \mathsf{p}(D,E) \wedge \mathsf{p}(E,F) \wedge \mathsf{f}(F),$$

which means having a granddaughter and a great-granddaughter but neither having a child who has a daughter and a granddaughter nor having a granddaughter who has a daughter. We call such a conjunction a *property itemset*.

3 Ideas and an Algorithm

We propose an algorithm based on MAPIX in order to produce rich combinations of properties. It keeps efficiency by seeing only combination appeared in samples.

In order to explain our idea, we introduce the shadow of a property item, which is a set of properties that produce the property item.

Definition 6. *For a database R and a property item it, the set defined bellow is called the shadow of the property item,*

$$\mathbf{shadow}(\mathsf{it}, R) = \{\, `e \leftarrow L' \in T \times 2^R \mid L \text{ is a property and } \mathbf{var}(e \leftarrow L) \sim \mathsf{it} \},$$

where T is the target relation in R, 2^X is the power set of a set X, and $P \sim Q$ means P and Q are θ-equivalent, i.e. P θ-subsumes Q and Q θ-subsumes P.

Example 4. For $R_{\mathbf{fam}}$ and $\mathsf{it2}$, the shadow of it is

$$\begin{array}{l}\mathbf{shadow}(\mathsf{it2}, R_{\mathbf{fam}}) = \{\\ \mathsf{gf}(01) \leftarrow \{\mathsf{p}(01,02), \mathsf{p}(02,03), \mathsf{f}(03)\},\ \mathsf{gf}(01) \leftarrow \{\mathsf{p}(01,02), \mathsf{p}(02,04), \mathsf{f}(04)\},\\ \mathsf{gf}(07) \leftarrow \{\mathsf{p}(07,08), \mathsf{p}(08,09), \mathsf{f}(09)\},\ \mathsf{gf}(12) \leftarrow \{\mathsf{p}(12,13), \mathsf{p}(13,15), \mathsf{f}(15)\},\\ \mathsf{gf}(12) \leftarrow \{\mathsf{p}(12,14), \mathsf{p}(14,16), \mathsf{f}(16)\},\ \mathsf{gf}(19) \leftarrow \{\mathsf{p}(19,20), \mathsf{p}(20,21), \mathsf{f}(21)\},\\ \mathsf{gf}(19) \leftarrow \{\mathsf{p}(19,20), \mathsf{p}(20,22), \mathsf{f}(22)\},\ \mathsf{gf}(20) \leftarrow \{\mathsf{p}(20,21), \mathsf{p}(21,24), \mathsf{f}(24)\}\ \ \}\end{array}$$

Definition 7 (combinable property itemset). *A property itemset* $\langle \mathsf{it}_{i_1}, \ldots, \mathsf{it}_{i_n} \rangle$ *is* combinable*, if there exist a target instance* e *and*

$$\langle e \leftarrow \mathsf{pr}_{i_1}, \ldots, e \leftarrow \mathsf{pr}_{i_n} \rangle \in \mathbf{shadow}(\mathsf{it}_{i_1}, R) \times \ldots \times \mathbf{shadow}(\mathsf{it}_{i_n}, R),$$

such that $\bigcap_{j=1,\ldots,n}(\mathbf{terms}(\mathsf{pr}_{i_j}) - \mathbf{terms}(e)) \neq \emptyset$*, where* $\mathbf{terms}(p)$ *is the set of all terms in* p*.* $\langle e \leftarrow \mathsf{pr}_{i_1}, \ldots, e \leftarrow \mathsf{pr}_{i_n} \rangle$ *is called a combinable shadow tuple.*

Definition 8 (combined property item). *When a property itemset* $\mathsf{is} = \langle \mathsf{it}_{i_1}, \ldots, \mathsf{it}_{i_n} \rangle$ *is combinable and* $\langle e \leftarrow \mathsf{pr}_{i_1}, \ldots, e \leftarrow \mathsf{pr}_{i_n} \rangle$ *is a combinable shadow tuple of it,*

$$\mathbf{var}(e \leftarrow \textstyle\bigcup_{j=1,\ldots,n} \mathsf{pr}_{i_j})$$

is called a combined property item produced from is*. A property item that is not combined is called* atomic*.*

Example 5. A property itemset $\langle \mathsf{it2}, \mathsf{it4} \rangle$ is combinable, because the shadow of $\mathsf{it2}$ (see the example above) and the shadow of $\mathsf{it4}$

$$\begin{array}{rl}\mathbf{shadow}(\mathsf{it4}, R_{\mathbf{fam}}) = \{\ & \mathsf{gf}(01) \leftarrow \{\mathsf{p}(01,02), \mathsf{p}(02,04), \mathsf{p}(04,06), \mathsf{f}(06)\},\\ & \mathsf{gf}(07) \leftarrow \{\mathsf{p}(07,08), \mathsf{p}(08,09), \mathsf{p}(09,11), \mathsf{f}(11)\},\\ & \mathsf{gf}(19) \leftarrow \{\mathsf{p}(19,20), \mathsf{p}(20,21), \mathsf{p}(21,24), \mathsf{f}(24)\}\ \}\end{array}$$

have five combinable shadow tuples

$$\begin{array}{l}\langle \mathsf{gf}(01) \leftarrow \{\mathsf{p}(01{,}02), \mathsf{p}(02{,}03), \mathsf{f}(03)\},\ \mathsf{gf}(01) \leftarrow \{\mathsf{p}(01{,}02), \mathsf{p}(02{,}04), \mathsf{p}(04{,}06), \mathsf{f}(06)\}\rangle,\\ \langle \mathsf{gf}(01) \leftarrow \{\mathsf{p}(01{,}02), \mathsf{p}(02{,}04), \mathsf{f}(04)\},\ \mathsf{gf}(01) \leftarrow \{\mathsf{p}(01{,}02), \mathsf{p}(02{,}04), \mathsf{p}(04{,}06), \mathsf{f}(06)\}\rangle,\\ \langle \mathsf{gf}(07) \leftarrow \{\mathsf{p}(07{,}08), \mathsf{p}(08{,}09), \mathsf{f}(09)\},\ \mathsf{gf}(07) \leftarrow \{\mathsf{p}(07{,}08), \mathsf{p}(08{,}09), \mathsf{p}(09{,}11), \mathsf{f}(11)\}\rangle,\\ \langle \mathsf{gf}(19) \leftarrow \{\mathsf{p}(19{,}20), \mathsf{p}(20{,}21), \mathsf{f}(21)\},\ \mathsf{gf}(19) \leftarrow \{\mathsf{p}(19{,}20), \mathsf{p}(20{,}21), \mathsf{p}(21{,}24), \mathsf{f}(24)\}\rangle,\\ \langle \mathsf{gf}(19) \leftarrow \{\mathsf{p}(19{,}20), \mathsf{p}(20{,}22), \mathsf{f}(22)\},\ \mathsf{gf}(19) \leftarrow \{\mathsf{p}(19{,}20), \mathsf{p}(20{,}21), \mathsf{p}(21{,}24), \mathsf{f}(24)\}\rangle\end{array}$$

and they produce the following two combined property items.

$$\begin{array}{rl}\mathsf{it2\text{-}4} &= \mathsf{gf}(A) \leftarrow \mathsf{p}(A,B) \wedge \mathsf{p}(B,C) \wedge \mathsf{f}(C) \wedge \mathsf{p}(C,D) \wedge \mathsf{f}(D).\\ \mathsf{it2\text{-}4'} &= \mathsf{gf}(A) \leftarrow \mathsf{p}(A,B) \wedge \mathsf{p}(B,C) \wedge \mathsf{f}(C) \wedge \mathsf{p}(B,D) \wedge \mathsf{p}(D,E) \wedge \mathsf{f}(E).\end{array}$$

$\mathsf{it2\text{-}4}$ means having a granddaughter who has a daughter and $\mathsf{it2\text{-}4'}$ means having a child who has a daughter and a granddaughter. Every tuple produces a combined property item equivalent to one of the above. All frequent combined property items produced from $R_{\mathbf{fam}}$ are shown in Table 2.

Table 2. All frequent property items and combined property items in $R_{\mathbf{fam}}$ for min sup=60%.

it0	$= \mathsf{gf}(A) \leftarrow \mathsf{m}(A).$
it1	$= \mathsf{gf}(A) \leftarrow \mathsf{p}(A,B) \wedge \mathsf{f}(B).$
it2	$= \mathsf{gf}(A) \leftarrow \mathsf{p}(A,B) \wedge \mathsf{p}(B,C) \wedge \mathsf{f}(C).$
it3	$= \mathsf{gf}(A) \leftarrow \mathsf{p}(A,B) \wedge \mathsf{p}(B,C) \wedge \mathsf{p}(C,D) \wedge \mathsf{m}(D).$
it4	$= \mathsf{gf}(A) \leftarrow \mathsf{p}(A,B) \wedge \mathsf{p}(B,C) \wedge \mathsf{p}(C,D), \mathsf{f}(D).$
it1-2	$= \mathsf{gf}(A) \leftarrow \mathsf{p}(A,B) \wedge \mathsf{f}(B) \wedge \mathsf{p}(B,C) \wedge \mathsf{f}(C).$
it1-3	$= \mathsf{gf}(A) \leftarrow \mathsf{p}(A,B) \wedge \mathsf{f}(B) \wedge \mathsf{p}(B,C) \wedge \mathsf{p}(C,D) \wedge \mathsf{m}(D).$
it2-3	$= \mathsf{gf}(A) \leftarrow \mathsf{p}(A,B) \wedge \mathsf{p}(B,C) \wedge \mathsf{f}(C) \wedge \mathsf{p}(C,D) \wedge \mathsf{m}(D).$
it2-3'	$= \mathsf{gf}(A) \leftarrow \mathsf{p}(A,B) \wedge \mathsf{p}(B,C) \wedge \mathsf{f}(C) \wedge \mathsf{p}(B,D) \wedge \mathsf{p}(D,E) \wedge \mathsf{m}(E).$
it2-4	$= \mathsf{gf}(A) \leftarrow \mathsf{p}(A,B) \wedge \mathsf{p}(B,C) \wedge \mathsf{f}(C) \wedge \mathsf{p}(C,D) \wedge \mathsf{f}(D).$
it2-4'	$= \mathsf{gf}(A) \leftarrow \mathsf{p}(A,B) \wedge \mathsf{p}(B,C) \wedge \mathsf{f}(C) \wedge \mathsf{p}(B,D) \wedge \mathsf{p}(D,E) \wedge \mathsf{f}(E).$
it3-4	$= \mathsf{gf}(A) \leftarrow \mathsf{p}(A,B) \wedge \mathsf{p}(B,C) \wedge \mathsf{p}(C,D) \wedge \mathsf{f}(D) \wedge \mathsf{p}(B,E) \wedge \mathsf{p}(E,F), \mathsf{m}(F).$
it1-2-3	$= \mathsf{gf}(A) \leftarrow \mathsf{p}(A,B) \wedge \mathsf{f}(B) \wedge \mathsf{p}(B,C) \wedge \mathsf{f}(C) \wedge \mathsf{p}(C,D) \wedge \mathsf{m}(D).$
it1-2-3'	$= \mathsf{gf}(A) \leftarrow \mathsf{p}(A,B) \wedge \mathsf{f}(B) \wedge \mathsf{p}(B,C) \wedge \mathsf{f}(C) \wedge \mathsf{p}(B,D) \wedge \mathsf{p}(D,E) \wedge \mathsf{m}(E).$
it2-3-4	$= \mathsf{gf}(A) \leftarrow \mathsf{p}(A,B) \wedge \mathsf{p}(B,C) \wedge \mathsf{f}(C) \wedge \mathsf{p}(C,D) \wedge \mathsf{f}(D) \wedge \mathsf{p}(B,E) \wedge \mathsf{p}(E,F) \wedge \mathsf{m}(F).$
it2-3-4'	$= \mathsf{gf}(A) \leftarrow \mathsf{p}(A,B) \wedge \mathsf{p}(B,C) \wedge \mathsf{p}(C,D) \wedge \mathsf{f}(D) \wedge \mathsf{p}(B,E) \wedge \mathsf{f}(E) \wedge \mathsf{p}(E,F) \wedge \mathsf{m}(F).$

With atomic property items combined property items work to make other patterns by conjunction. For example, it1 and it2-4 make the following pattern,

$$\langle \mathsf{it1}, \mathsf{it2\text{-}4} \rangle = \mathsf{gf}(A) \leftarrow \mathsf{p}(A,B) \wedge \mathsf{f}(B) \wedge \mathsf{p}(A,C) \wedge \mathsf{p}(C,D) \wedge \mathsf{f}(D) \wedge \mathsf{p}(D,E) \wedge \mathsf{f}(E),$$

which means having a daughter and a granddaughter who has a daughter. □

Here we propose a new algorithm for all frequent patterns made from conjunction among atomic and combined property items extracted from samples:

1. It samples target instances from a target relation.
2. For each sampled instance it extracts atomic property items hold on DB.
3. By using an Apriori-like level-wise algorithm it enumerates all frequent conjunctions of the atomic property items.
4. It produces all combined property items from the frequent combinable conjunction.
5. Again by the level-wise algorithm it enumerates all frequent conjunctions of atomic and combined property items.

The detail of the algorithm is given in Table 3.

4 Experiments and Concluding Remarks

We have done two experiments. The first one was with $R_{\mathbf{fam}}$, where we aim to see the varieties of patterns extracted. Table 3 shows the numbers of patterns enumerated and runtime for our algorithm as well as MAPIX and the algorithm in [4] according as the minimum support threshold is changed. Our algorithm produced more patterns than others. There was no duplication in the patterns enumerated by three algorithms in the sense of θ-equivalence.

Table 3. A proposing algorithm.

input $\boldsymbol{R}$: a DB; T : target relation; $\sup_{\min}$: the min. sup. threshold;
output *Freq*: the set of patterns whose supports are larger than $\sup_{\min}$

1. **Select** an appropriate size of subset $T' \subseteq T$;
2. $Items := \emptyset$; $\mathcal{P} := \emptyset$; $Freq := \emptyset$;
3. **For each** $e \in T'$ **do** $\mathcal{P} := \mathcal{P} \cup \{e \leftarrow \mathsf{pr} \mid \mathsf{pr}$ is a property of $e\}$
4. **For each** '$e \leftarrow \mathsf{pr}$' $\in \mathcal{P}$ **do**
5. **If** $\exists I \in Items$, $I \sim \mathbf{var}(e \leftarrow \mathsf{pr})$ **then** $S[I] := S[I] \cup \{e \leftarrow \mathsf{pr}\}$;
6. **else** $I' = \mathbf{var}(e \leftarrow \mathsf{pr})$; $S[I'] := \{e \leftarrow \mathsf{pr}\}$; $Items := Items \cup \{I'\}$;
7. $k := 1$; $\mathcal{F}_1^1 := \{\langle I \rangle \mid I \in Items$ and $\mathbf{supp}(I) \geq \sup_{\min}\}$; $Freq := \mathcal{F}_1^1$;
8. **While** $\mathcal{F}_k^1 \neq \emptyset$ **do**
9. $\mathcal{C}_{k+1} :=$ CANDIDATE$(\mathcal{F}_k^1, \mathcal{F}_k^1)$; $\mathcal{F}_{k+1}^1 := \{IS \in \mathcal{C}_{k+1} \mid \mathbf{supp}(IS) \geq \sup_{\min}\}$;
10. $Freq := Freq \cup \mathcal{F}_{k+1}^1$; $k := k+1$;
11. $Combined :=$ CANDICOMB$(Freq)$;
12. $k := 1$; $\mathcal{F}_1^2 := \{\langle I \rangle \mid I \in Combined$ and $\mathbf{supp}(I) \geq \sup_{\min}\}$; $Freq := Freq \cup \mathcal{F}_1^2$;
13. **While** $\mathcal{F}_k^2 \neq \emptyset$ **do**
14. $\mathcal{C}_{k+1} :=$ CANDIDATE$(\mathcal{F}_k^1, \mathcal{F}_k^2)$; $\mathcal{F}_{k+1}^2 := \{IS \in \mathcal{C}_{k+1} \mid \mathbf{supp}(IS) \geq \sup_{\min}\}$;
15. $Freq := Freq \cup \mathcal{F}_{k+1}^2$; $k := k+1$;
16. **Return** *Freq*;

CANDIDATE$(\mathcal{F}_k^1, \mathcal{F}_k^2)$:
input $\mathcal{F}_k^1, \mathcal{F}_k^2$: sets of frequent property itemsets of a level;
output $\mathcal{C}_{k+1}$: the set of candidate property itemsets of the next level where at least a property itemset is used from $\mathcal{F}_k^2$;

1. $\mathcal{C}_{k+1} := \emptyset$
2. **For each** pair $\langle\langle I_1, \ldots, I_k\rangle, \langle I'_1, \ldots, I'_k\rangle\rangle \in \mathcal{F}_k^2 \times \{\mathcal{F}_k^1 \cup \mathcal{F}_k^2\}$ where $I_1 = I'_1, \ldots, I_{k-1} = I'_{k-1}$, and $I_k < I'_k$ **do**
3. $\mathcal{C}_{k+1} := \mathcal{C}_{k+1} \cup \{\langle I_1, \ldots, I_{k-1}, I_k, I'_k\rangle\}$;
4. **For each** $IS \in \mathcal{C}_{k+1}$ **do**
5. **If** $k = 1$ and $(I \preceq I'$ or $I' \preceq I)$, where $IS = \langle I, I'\rangle$ **then remove** IS from $\mathcal{C}_{k+1}$;
6. **For each** $I \in IS$ **do if** $IS - \{I\} \notin \mathcal{F}_k^1 \cup \mathcal{F}_k^2$ **then remove** IS from $\mathcal{C}_{k+1}$;
7. **Return** $\mathcal{C}_{k+1}$;

CANDICOMB(*Freq*):
input *Freq* : the set of frequent property item sets made of atomic items;
output *Combined* : the set of combined property items produced from all property itemsets in *Freq*;

1. $Combined := \emptyset$;
2. **For each** $\langle I_1, \ldots, I_n\rangle \in Freq$ for $n \geq 2$ **do**
3. **For each** $\langle e_1 \leftarrow \mathsf{pr}_1, \ldots, e_n \leftarrow \mathsf{pr}_n\rangle \in S[I_1] \times \cdots \times S[I_n]$ **do**
4. **If** $e_1 = \ldots = e_n$ and $\bigcap_{j=1,\ldots,n}(\mathbf{terms}(\mathsf{pr}_j) - \mathbf{terms}(e_j)) \neq \emptyset$ **then**
5. $Combined := Combined \cup \{\mathbf{var}(e_1 \leftarrow \bigcup_{j=1,\ldots,n} \mathsf{pr}_j)\}$;
6. **For each** $I \in Combined$ **do**
7. **If** $\exists I' \in Combined$, $I' \neq I \wedge I' \sim I$ **then** $Combined := Combined - \{I\}$;
6. **Return** *Combined*;

The second experiment was with the data of Bongard. Fig. 2 shows the numbers of patterns and runtime when the number of examples to extract properties

Table 3. Experiment with $R_{\mathbf{fam}}$. All examples were used.

	min sup	20%	40%	60%	80%
MAPIX	#patterns	55	31	23	11
[5]	time (sec)	0.04	0.01	0.01	0.01
algo.	#patterns	0.44	0.15	0.05	0.02
in [4]	time (sec)	6.23	0.45	0.06	0.03
our	#patterns	4601	1063	109	17
algo.	time (sec)	9.55	0.54	0.08	0.01

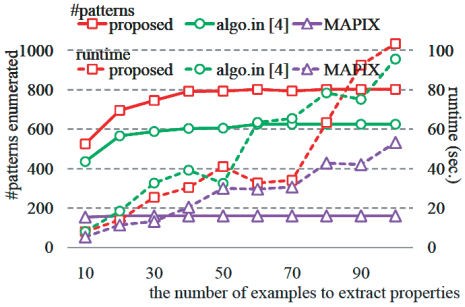


Fig. 2. The numbers of patterns and runtime as the number of sampled examples increase in Bongard.

changes. These are the average of 10 times execution. We used all examples to count frequency but sampled limited number examples to extract properties. By 80 examples our algorithm produced the same set of patterns (802 patterns) as the case using whole 392 examples. The 802 patterns had no duplication. Among 802 patterns 99 have at most 5 literals, 533 have 6 to 10 literals, 145 have 11 to 15 literals and 25 are longer than 15. The longest pattern has 19 literals. Fig. 2 also shows the grow of runtime according to the sample size.

Our algorithm enumerates larger range of patterns, made of property items and combined property items in which property items are combined as in examples. The algorithm in [4] uses structural combination but it has limitation. It treats property items only in a single example then it composes all examples as an artificial large example. In stead of our shadow the algorithm keeps the conjunction of all equivalent property items because the conjunction is equivalent to each property item. By the method patterns becomes larger. Our algorithm overcome these limitation by the shadow and simple algorithm, but has the large time complexity to the size of examples sampled. In fact it took 6556 seconds for Bongard when it samples all examples. The sufficient size of examples, however, can be suppressed because it depends only on the minimum support threshold.

References

1. R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. VLDB, pp. 487–499, 1994.
2. L. Dehaspe and L. De Raedt. Mining association rules with multiple relations. ILP97, pp.125–132, 1997.
3. L. Dehaspe, and H. Toivonen. Discovery of frequent Datalog patterns. Data Mining and Knowledge Discovery, Vol. 3, no. 1, pp. 7-36, 1999.
4. N. Inuzuka, J. Motoyama, S. Urazawa and T. Nakano. Relational pattern mining based on equivalent classes of properties extracted from samples. PAKDD2008, pp. 582–591, 2008.
5. J. Motoyama, S. Urazawa, T. Nakano and N. Inuzuka. A mining algorithm using property items extracted from sampled examples, ILP2006, pp.335–350, 2007.