



**HAL**  
open science

## Fast Generation of Best Interval Patterns for Nonmonotonic Constraints

Aleksey Buzmakov, Sergei O. Kuznetsov, Amedeo Napoli

► **To cite this version:**

Aleksey Buzmakov, Sergei O. Kuznetsov, Amedeo Napoli. Fast Generation of Best Interval Patterns for Nonmonotonic Constraints. Machine Learning and Knowledge Discovery in Databases, Sep 2015, Porto, Portugal. pp.157-172, 10.1007/978-3-319-23525-7\_10 . hal-01186718

**HAL Id: hal-01186718**

**<https://hal.science/hal-01186718>**

Submitted on 25 Aug 2015

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Fast Generation of Best Interval Patterns for Nonmonotonic Constraints\*

Aleksey Buzmakov<sup>1,2</sup>      Sergei O. Kuznetsov<sup>2</sup>  
Amedeo Napoli<sup>1</sup>

<sup>1</sup>LORIA (CNRS – Inria NGE – U. de Lorraine),  
Vandœuvre-lès-Nancy, France

<sup>2</sup>National Research University Higher School of Economics,  
Moscow, Russia  
aleksey.buzmakov@inria.fr, skuznetsov@hse.ru,  
amedeo.napoli@loria.fr

## Abstract

In pattern mining, the main challenge is the exponential explosion of the set of patterns. Typically, to solve this problem, a constraint for pattern selection is introduced. One of the first constraints proposed in pattern mining is support (frequency) of a pattern in a dataset. Frequency is an anti-monotonic function, i.e., given an infrequent pattern, all its superpatterns are not frequent. However, many other constraints for pattern selection are neither monotonic nor anti-monotonic, which makes it difficult to generate patterns satisfying these constraints. In this paper we introduce the notion of “generalized monotonicity” and  $\Sigma\phi\alpha$  algorithm that allow generating best patterns in polynomial time for some nonmonotonic constraints modulo constraint computation and pattern extension operations. In particular, this algorithm is polynomial for data on itemsets and interval tuples. In this paper we consider stability and  $\Delta$ -measure which are nonmonotonic constraints and apply them to interval tuple datasets. In the experiments, we compute best interval tuple patterns w.r.t. these measures and show the advantage of our approach over postfiltering approaches.

**KEYWORDS:** Pattern mining, nonmonotonic constraints, interval tuple data

## 1 Introduction

Interestingness measures were proposed to overcome the problem of combinatorial explosion of the number of valid patterns that can be discovered in a

---

\*The final publication is available at [link.springer.com](http://link.springer.com)

dataset [17]. For example, pattern support, i.e., the number of objects covered by the pattern, is one of the most famous measures of pattern quality. In particular, support satisfies the property of anti-monotonicity (aka “a priori principle”), i.e., the larger the pattern is the smaller the support is [11, 1]. Many other measures can be mentioned such as utility constraint [23], pattern stability [9, 14], pattern leverage [18], margin closeness [12], MCCS [15], cosine interest [4], pattern robustness [16], etc.

Some of these measures (e.g., support, robustness for generators [16], or upper bound constraint of MCCS [15]) are “globally anti-monotonic”, i.e., for any two patterns  $X \sqsubseteq Y$  we have  $\mathcal{M}(X) \geq \mathcal{M}(Y)$ , where  $\mathcal{M}$  is a measure. When a measure is anti-monotonic, it is relatively easy to find patterns whose measure is higher than a certain threshold (e.g. patterns with a support higher than a threshold). In contrast some other measures are called “locally anti-monotonic”, i.e., for any pattern  $X$  there is an immediate subpattern  $Y \prec X$  such that  $\mathcal{M}(Y) \geq \mathcal{M}(X)$ . Then the right strategy should be selected for traversing the search space, e.g. a pattern  $Y$  should be extended only to patterns  $X$  such that  $\mathcal{M}(X) \leq \mathcal{M}(Y)$ . For example, for “locally anti-monotonic” cosine interest [4], the extension of a pattern  $Y$  consists in adding only attributes with a smaller support than any attribute from  $Y$ . The most difficult case for selecting valid patterns occurs when a measure is not locally anti-monotonic. Then, valid patterns can be retained by postfiltering, i.e., finding a (large set of) patterns satisfying an antimonotone constraint and filtering them w.r.t. the chosen nonmonotonic measure (i.e. neither monotonic nor anti-monotonic) [14, 12, 16], or using heuristics such as leap search [21] or low probability of finding interesting patterns in the current branch [18].

Most of the measures are only applicable to one type of patterns, e.g., pattern leverage or cosine interest can be applied only to binary data since their definitions involve single attributes. “Pattern independent measures” usually relies on support of the pattern and/or on support of other patterns from the search space. In particular, support, stability [9], margin-closeness [12] and robustness [16] are pattern independent measures. In this paper we work with interval tuple data, where only pattern independent measures as well as specific measures for interval tuples can be applied. In addition, given a measure, it can be difficult to define a good threshold. Thus various approaches for finding  $K$ -top patterns were introduced [7, 20, 19], with the basic idea to automatically adjust the threshold for a measure  $\mathcal{M}$ .

In this paper we introduce a new algorithm  $\Sigma\omicron\phi\iota$ , i.e. Sofia, for “Searching for Optimal Formal Intents Algorithm”, for extracting the best patterns of a kind, e.g., itemsets, interval tuples, strings, graph patterns, etc.  $\Sigma\omicron\phi\iota$  algorithm is applicable to a class of measures called “projection monotonic measures” or more precisely “measures monotonic w.r.t. a chain of projections”. This class includes globally anti-monotonic measures such as support, locally anti-monotonic measures such as cosine interest and some of the nonmonotonic measures such as stability or robustness of closed patterns. We also introduce a way of adjusting a measure threshold such that the number of generated patterns is limited. This allows finding the best patterns w.r.t. a projection monotonic

measure in polynomial time modulo complexity of basic operations (i.e. intersecting and testing containment on descriptions, computation of a measure). The main novelty of this paper is  $\Sigma\phi\alpha$ , a new efficient polynomial algorithm for finding best patterns of different kinds w.r.t. projection monotonic measures which constitutes a rather large class of measures.

The remaining of the paper is organized as follows. The formalization of the current approach is based on Formal Concept Analysis (FCA) [6] and pattern structures [5] which are introduced in Section 2. Then,  $\Sigma\phi\alpha$  algorithm is detailed in Section 3 first for an arbitrary measure and second for the  $\Delta$ -measure. Experiments and a discussion are proposed in Section 4, before conclusion.

## 2 Data Model

### 2.1 FCA and Pattern structures

Formal Concept Analysis (FCA) is a formalism for modeling knowledge discovery and data mining thanks to the design of concept lattices [6]. It is also convenient for describing models of itemset mining, and, since [13], lattices of closed itemsets (i.e. concept lattices) and closed descriptions are used for concise representation of association rules. For more complex data such as sequences and graphs one can use an extension of the basic model, called pattern structures [5]. With pattern structures it is possible to define closed descriptions and to give a concise representation of association rules for different descriptions with a natural order (such as subgraph isomorphism order) [10, 8].

A *pattern structure* is a triple  $(G, (D, \sqcap), \delta)$ , where  $G$  is a set of objects,  $(D, \sqcap)$  is a complete meet-semilattice of descriptions and  $\delta : G \rightarrow D$  maps an object to a description.

The intersection  $\sqcap$  gives similarity of two descriptions. Standard FCA can be presented in terms of a pattern structure. A *formal context*  $(G, M, I)$ , where  $G$  is a set of objects,  $M$  is a set of attributes and  $I \subseteq G \times M$  an incidence relation giving information about attributes related to every object, is represented as a pattern structure  $(G, (\wp(M), \cap), \delta)$ , where  $(\wp(M), \cap)$  is a semilattice of subsets of  $M$  with  $\cap$  being set-theoretical intersection. If  $x = \{a, b, c\}$  and  $y = \{a, c, d\}$ , then  $x \sqcap y = x \cap y = \{a, c\}$ . The mapping  $\delta : G \rightarrow \wp(M)$  is given by  $\delta(g) = \{m \in M \mid (g, m) \in I\}$  and returns the description of a given object as a set of attributes.

The following mappings or diamond operators give a connection between the powerset of objects and descriptions:

$$A^\diamond := \bigsqcap_{g \in A} \delta(g), \quad \text{for } A \subseteq G$$

$$d^\diamond := \{g \in G \mid d \sqsubseteq \delta(g)\}, \quad \text{for } d \in D$$

Given a subset of objects  $A$ ,  $A^\diamond$  returns the description which is common to all objects in  $A$ . Given a description  $d$ ,  $d^\diamond$  is the set of all objects whose description subsumes  $d$ . A partial order  $\sqsubseteq$  (subsumption) on descriptions from

$D$  is defined w.r.t. the similarity operation  $\sqcap$ :  $c \sqsubseteq d \Leftrightarrow c \sqcap d = c$ , and  $c$  is subsumed by  $d$ .

A *pattern concept* of a pattern structure  $(G, (D, \sqcap), \delta)$  is a pair  $(A, d)$ , where  $A \subseteq G$ , called *pattern extent* and  $d \in D$ , called *pattern intent*, such that  $A^\diamond = d$  and  $d^\diamond = A$ . Pattern extent is a closed set of objects, and a pattern intent is a closed description, e.g., a closed itemset when descriptions are given as sets of items (attributes). As shown in [10], descriptions closed in terms of counting inference (which is a standard data mining approach), such as closed graphs [22], are elements of pattern intents.

A pattern extent corresponds to the maximal set of objects  $A$  whose descriptions subsume the description  $d$ , where  $d$  is the maximal common description for objects in  $A$ . The set of all pattern concepts is partially ordered w.r.t. inclusion on extents, i.e.,  $(A_1, d_1) \leq (A_2, d_2)$  iff  $A_1 \subseteq A_2$  (or, equivalently,  $d_2 \sqsubseteq d_1$ ), making a lattice, called pattern lattice.

## 2.2 Interval pattern structure

Another possible instantiation of pattern structures is interval pattern structures introduced to support efficient processing of numerical data without binarization [8]. Given  $k$  numerical or interval attributes whose values are of the form  $[a, b]$ , where  $a, b \in \mathbb{R}$ , the language of a pattern space is given by tuples of intervals of size  $k$ . For simplicity, we denote intervals of the form  $[a, a]$  by  $a$ .

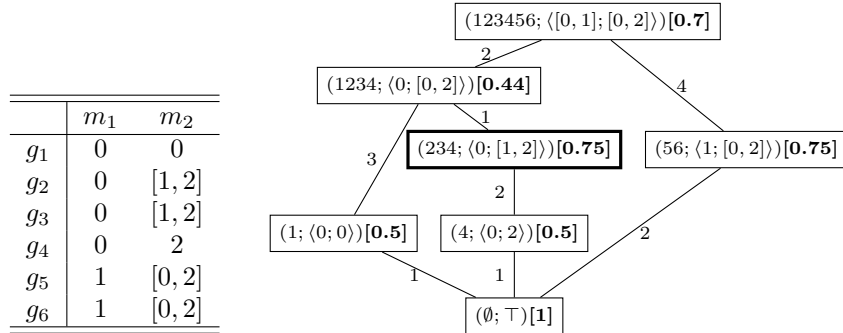
Figure 1a exemplifies an interval dataset. It contains 6 objects and 2 attributes. An interval as a value of an attribute corresponds to an uncertainty in the value of the attribute. For example, the value of  $m_1$  of  $g_2$  is known exactly, while the value of  $m_2$  is somewhere in  $[1, 2]$ . Given this intuition for the intervals it is natural to define similarity of two intervals as their convex hull, since by adding new objects one increases the uncertainty. For example, for  $g_1$  the value of  $m_1$  is 0, while for  $g_6$  is 1, thus given the set  $\{g_1, g_6\}$ , the uncertainty of  $m_1$  in this set is  $[0, 1]$ , i.e. the similarity of  $g_1$  and  $g_6$  w.r.t.  $m_1$  is  $[0, 1]$ . More formally, given two intervals  $[a, b]$  and  $[c, d]$ , the similarity of these two intervals is given by  $[a, b] \sqcap [c, d] = [\min(a, c), \max(b, d)]$ . Given a tuple of intervals, the similarity is computed component-wise. For example,  $g_1^\diamond \sqcap g_6^\diamond = \langle [0, 1]; [0, 2] \rangle$ .

The resulting concept lattice is shown in Figure 1b. Concept extents are shown by indices of objects, intents are given in angle brackets, the numbers on edges and on concepts are related to interestingness of concepts and will be described in the next subsection.

## 2.3 Stability index of a concept

For real datasets, the number of patterns can be large. Different measures were tested for selecting most interesting patterns, such as stability [9]. Stability measures the independence of a concept intent w.r.t. randomness in data.

Given a concept  $\mathcal{C}$ , *concept stability*  $\text{Stab}(\mathcal{C})$  is the relative number of subsets of the concept extent (denoted  $\text{Ext}(\mathcal{C})$ ), whose description, i.e. the result of  $(\cdot)^\diamond$



(a) An interval context. (b) An interval concept lattice with corresponding stability indexes. Objects are given by their indices.

Figure 1: A formal context and the corresponding lattice.

is equal to the concept intent (denoted  $\text{Int}(\mathcal{C})$ ).

$$\text{Stab}(\mathcal{C}) := \frac{|\{s \in \wp(\text{Ext}(\mathcal{C})) \mid s^\diamond = \text{Int}(\mathcal{C})\}|}{|\wp(\text{Ext}(\mathcal{C}))|} \quad (1)$$

Here  $\wp(P)$  is the powerset of  $P$ . The larger the stability, the more objects can be deleted from the context without affecting the intent of the concept, i.e. the intent of the most stable concepts is likely to be a characteristic pattern of a given phenomenon and not an artifact of a dataset.

We say that a concept is stable if stability of it is higher than a given threshold  $\theta$ ; a pattern  $p$  is stable if there is a concept in the lattice with  $p$  as the intent and the concept is stable.

**Example 1** Figure 1b shows a lattice for the context in Figure 1a. Concept extents are given by their indices, i.e.,  $\{g_1, g_2\}$  is given by 12. The extent of the highlighted concept  $\mathcal{C}$  is  $\text{Ext}(\mathcal{C}) = \{g_2, g_3, g_4\}$ , thus, its powerset contains  $2^3$  elements. Descriptions of 2 subsets of  $\text{Ext}(\mathcal{C})$  ( $\{g_4\}$  and  $\emptyset$ ) are different from  $\text{Int}(\mathcal{C}) = \{m_3\}$ , while all other subsets of  $\text{Ext}(\mathcal{C})$  have a common set of attributes equal to  $\langle 0; [1, 2] \rangle$ . So,  $\text{Stab}(\mathcal{C}) = \frac{2^3 - 2}{2^3} = 0.75$ . Stability of other concepts is shown in brackets. It should be noticed, that stability of all comparable patterns for  $\text{Int}(\mathcal{C})$  in the lattice are smaller than the stability of  $\mathcal{C}$ , which highlights the nonmonotonicity of stability.

Concept stability is closely related to the robustness of a closed pattern [16]. Indeed, robustness is the probability of a closed pattern to be found in a subset of the dataset. To define this probability, the authors define a weight for every subset given as a probability of getting this subset by removing objects from the dataset, where every object is removed with probability  $\alpha$ , e.g., given a subset of objects  $X \subseteq G$ , the probability of the induced subset is given by  $p(D_\alpha = (X, (D, \sqcap), \delta)) = \alpha^{|X|}(1 - \alpha)^{|G \setminus X|}$ . Stability in this case is the robustness of closed pattern if the weights of subsets of the dataset are equal to  $2^{-|G|}$ .

The problem of computing concept stability is #P-complete [9]. A fast computable stability estimate was proposed in [3], where it was shown that this estimate ranks concepts almost in the same way as stability does. In particular,  $\text{Stab}(\mathcal{C}) \leq 1 - 2^{-\Delta(\mathcal{C})}$ , where  $\Delta(\mathcal{C}) = \min_{\mathcal{D} \leq \mathcal{C}} |\text{Ext}(\mathcal{C}) \setminus \text{Ext}(\mathcal{D})|$ , i.e., the minimal difference in supports between concept  $\mathcal{C}$  and all its nearest subconcepts. For a threshold  $\theta$ , patterns  $p$  with  $\Delta(p) \geq \theta$  are called  $\Delta$ -stable patterns.

**Example 2** Consider the example in Figure 1. Every edge in the figure is labeled with the difference in support between the concepts this edge connects. Thus,  $\Delta$  of a pattern is the minimum label of the edges coming down from the concept. The value  $\Delta(\langle \{g_2, g_3, g_4\}; \langle 0; [1, 2] \rangle \rangle)$  is equal to 2. Another example is  $\Delta(\langle G; \langle [0, 1]; [0, 2] \rangle \rangle) = 2$ . For this example we can also see that  $\Delta$ -measure is not monotonic either.

$\Delta$ -measure is related to the work of margin-closeness of an itemset [12]. In this work, given a set of patterns, e.g., frequent closed patterns, the authors rank them by the minimal distance in their support to closest superpattern divided over the support of the pattern. In our case, the minimal distance is exactly the  $\Delta$ -measure of the pattern.

## 2.4 Projections of Pattern Structures

The approach proposed in this paper is based on projections introduced for reducing complexity of computing pattern lattices [5].

A projection  $\psi : D \rightarrow D$  is an “interior operator”, i.e. it is (1) monotonic ( $x \sqsubseteq y \Rightarrow \psi(x) \sqsubseteq \psi(y)$ ), (2) contractive ( $\psi(x) \sqsubseteq x$ ) and (3) idempotent ( $\psi(\psi(x)) = \psi(x)$ ). A projected pattern structure  $\psi(\langle G, (D, \sqcap), \delta \rangle)$  is a pattern structure  $\langle G, (D_\psi, \sqcap_\psi), \psi \circ \delta \rangle$ , where  $D_\psi = \psi(D) = \{d \in D \mid \exists d^* \in D : \psi(d^*) = d\}$  and  $\forall x, y \in D, x \sqcap_\psi y := \psi(x \sqcap y)$ .

**Example 3** Consider the example in Figure 1. If we remove a column corresponding to an attribute, e.g., the attribute  $m_2$ , from the context in Figure 1a, we define a projection, given by  $\psi(\langle [a, b]; [c, d] \rangle) = \langle [a, b]; [-\text{inf}, +\text{inf}] \rangle$ .

Given a projection  $\psi$  we call  $\psi(D) = \{d \in D \mid \psi(d) = d\}$  the fixed set of  $\psi$ . Note that, if  $\psi(d) \neq d$ , then there is no other  $\tilde{d}$  such that  $\psi(\tilde{d}) = d$  because of idempotency of projections. Hence, any element outside the fixed set of the projection  $\psi$  is pruned from the description space. Given the notion of a fixed set we can define a partial order on projections.

**Definition 1** Given a pattern structure  $\mathbb{P} = \langle G, (D, \sqcap), \delta \rangle$  and two projections  $\psi_1$  and  $\psi_2$ , we say that  $\psi_1$  is simpler than  $\psi_2$  ( $\psi_2$  is more detailed than  $\psi_1$ ), denoted by  $\psi_1 < \psi_2$ , if  $\psi_1(D) \subset \psi_2(D)$ , i.e.,  $\psi_1$  prunes more descriptions than  $\psi_2$ .

Our algorithm is based on this order on projections. The simpler a projection  $\psi$  is, the less patterns we can find in  $\psi(\mathbb{P})$ , and the less computational efforts

one should take. Thus, we compute a set of patterns for a simpler projection, then we remove unpromising patterns and extend our pattern structure and the found patterns to a more detailed projection. This allows us to reduce the size of patterns within a simpler projection in order to reduce the computational complexity of more detailed projection.

## 2.5 Projections of Interval Pattern Structures

Let us first consider interval pattern structures with only one attribute  $m$ . Let us denote by  $W = \{w_1, \dots, w_{|W|}\}$  all possible values of the left and right endpoints of the intervals corresponding to the attribute in a dataset, so that  $w_1 < w_2 < \dots < w_{|W|}$ . By reducing the set  $W$  of possible values for the left or the right end of the interval we define a projection. For example, if  $\{w_1\}$  is the only possible value for the left endpoint of an interval and  $\{w_{|W|}\}$  is the only possible value of the right endpoint of an interval, then all interval patterns are projected to  $[w_1, w_{|W|}]$ . Let us consider this in more detail.

Let two sets  $L, R \subset W$  such that  $w_1 \in L$  and  $w_{|W|} \in R$  be constraints on possible values on the left and right endpoints of an interval, respectively. Then a projection is defined as follows:

$$\psi_{m[L,R]}([a, b]) = [\max\{l \in L \mid l \leq a\}, \min\{r \in R \mid r \geq b\}]. \quad (2)$$

Requiring that  $w_1 \in L$  and  $w_{|W|} \in R$  we ensure that the sets used for minimal and maximal functions are not empty. It is not hard to see that (2) is a projection. The projections given by (2) are ordered w.r.t. simplicity (Definition 1). Indeed, given  $L_1 \subseteq L$  and  $R_1 \subseteq R$ , we have  $\psi_{m[L_1, R_1]} < \psi_{m[L, R]}$ , because of inclusion of fixed sets. Let us notice that a projection  $\psi_{m[W, W]}$  does not modify the lattice of concepts for the current dataset, since we forbid no interval that can be built for the value set  $W$ . Moreover, we can apply the projections of type (2) independently to every attribute of a tuple of intervals and the order on projections for tuples can be defined component-wise.

**Example 4** Consider example in Figure 1. Let us consider a projection

$$\psi_{m_1[\{0,1\}, \{1\}]m_2[\{0,2\}, \{0,2\}]}$$

The fixed set of this projection consists of  $\{[0, 1], 1\} \times \{0, 2, [0, 2]\}$ , i.e., 6 intervals. Let us find the projection of  $(g_2)^\diamond = \langle 0; [1, 2] \rangle$  in a component-wise way:  $\psi_{m_1[\{0,1\}, \{1\}]}(0) = [0, 1]$ , since 0 is allowed on the left endpoint of an interval but not allowed to be on the right endpoint of an interval;  $\psi_{m_2[\{0,2\}, \{0,2\}]}([1, 2]) = [0, 2]$  since 1 is not allowed on the left endpoint of an interval. Thus,

$$\psi_{m_1[\{0,1\}, \{1\}]m_2[\{0,2\}, \{0,2\}]}(\langle 0; [1, 2] \rangle) = \langle [0, 1]; [0, 2] \rangle.$$

The lattice corresponding to this projection is shown in Figure 2.



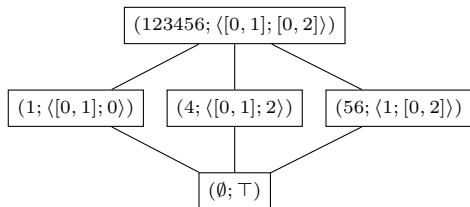


Figure 2: Projected lattice from example in Figure 1 by projection  $\psi_{m_1\{\{0,1\},\{1\}\}m_2\{\{0,2\},\{0,2\}\}}$ . See Example 4.

### 3 Σοφια Algorithm

#### 3.1 Monotonicity w.r.t. a Projection

Our algorithm is based on the projection monotonicity, a new idea introduced in this paper. Many interestingness measures for patterns, e.g., stability, are not (anti-)monotonic w.r.t. subsumption order on patterns. A measure  $\mathcal{M}$  is called *anti-monotonic*, if for two patterns  $q \sqsubseteq p$ ,  $\mathcal{M}(q) \geq \mathcal{M}(p)$ . For simplicity in the rest of the paper we skip the prefix “anti” and call such property monotonicity. For instance, support is a monotonic measure w.r.t. pattern order and it allows for efficient generation of patterns with support larger than a threshold [1, 11, 13]. The projection monotonicity is a generalization of standard monotonicity and allows for efficient work with a larger set of interestingness measures.

**Definition 2** *Given a pattern structure  $\mathbb{P}$  and a projection  $\psi$ , a measure  $\mathcal{M}$  is called monotonic w.r.t. the projection  $\psi$ , if*

$$(\forall p \in \psi(\mathbb{P}))(\forall q \in \mathbb{P}, \psi(q) = p) \mathcal{M}_\psi(p) \geq \mathcal{M}(q), \quad (3)$$

where  $\mathcal{M}_\psi$  is the measure  $\mathcal{M}$  of pattern  $p$  computed in  $\psi(\mathbb{P})$ .

Here, for any pattern  $p$  of a projected pattern structure we check that a preimage  $q$  of  $p$  for  $\psi$  has a measure smaller than the measure of  $p$ . It should be noticed that a measure  $\mathcal{M}$  for a pattern  $p$  can yield different values if  $\mathcal{M}$  is computed in  $\mathbb{P}$  or in  $\psi(\mathbb{P})$ . Thus we use the notation  $\mathcal{M}_\psi$  for the measure  $\mathcal{M}$  computed in  $\psi(\mathbb{P})$ . The property of a measure given in Definition 2 is called projection monotonicity.

It should be noticed that a classical monotonic measures are monotonic w.r.t. any projection. Indeed, because of contractivity of  $\psi$  ( $\psi(p) \sqsubseteq p$ ), for any anti-monotonic measure one has  $\mathcal{M}(\psi(p)) \geq \mathcal{M}(p)$ . This definition covers also the cases where a measure  $\mathcal{M}$  is only locally monotonic, i.e., given a pattern  $p$  there is an immediate subpattern  $q \prec p$  such that  $\mathcal{M}(q) \geq \mathcal{M}(p)$ , see e.g. the cosine interest of an itemset, which is only locally monotonic [4]. Moreover, this definition covers also some measures that are not locally monotonic. As we mentioned in Examples 1 and 2 stability and  $\Delta$ -measure are not locally monotonic. However, it can be shown that they are monotonic w.r.t.

any projection [2]. Moreover, following the same strategy one can prove that robustness of closed patterns from [16] is also monotonic w.r.t. any projection. In particular, the robustness of closed patterns defines a monotonic constraint w.r.t. any projection.

Thus, given a measure  $\mathcal{M}$  monotonic w.r.t. a projection  $\psi$ , if  $p$  is a pattern such that  $\mathcal{M}_\psi(p) < \theta$ , then  $\mathcal{M}(q) < \theta$  for any preimage  $q$  of  $p$  for  $\psi$ . Hence, if, given a pattern  $p$  of  $\psi(\mathbb{P})$ , one can find all patterns  $q$  of  $\mathbb{P}$  such that  $\psi(q) = p$ , it is possible to find the patterns of  $\psi(\mathbb{P})$  and then to filter them w.r.t.  $\mathcal{M}_\psi$  and a threshold, and finally to compute the preimages of filtered patterns. It allows one to cut earlier unpromising branches of the search space or adjust a threshold for finding only a limited number of best patterns.

### 3.2 Monotonicity w.r.t. a Chain of Projections

However, given just one projection, it can be hard to efficiently discover the patterns, because the projection is either hard to compute or the number of unpromising patterns that can be pruned is not high. Hence we introduce a chain of projections  $\psi_0 < \psi_1 < \dots < \psi_k = \mathbb{1}$ , where a pattern lattice for  $\psi_0(\mathbb{P})$  can be easily computed and  $\mathbb{1}$  is the identity projection, i.e.,  $(\forall x)\mathbb{1}(x) = x$ . For example, to find frequent itemsets, we typically search for small frequent itemsets and then extend them to larger ones. It corresponds to extension to a more detailed projection.

**Definition 3** *Given a pattern structure  $\mathbb{P}$  and a chain of projections  $\psi_0 < \psi_1 < \dots < \psi_k = \mathbb{1}$ , a measure  $\mathcal{M}$  is called monotonic w.r.t. the chain of projections if  $\mathcal{M}$  is monotonic w.r.t. all  $\psi_i$  for  $0 \leq i \leq k$ .*

**Example 5** *Let us construct a chain of projections satisfying (2) for the example in Figure 1. The value set for the first attribute is  $W_1 = \{0, 1\}$  and the value set for the second is  $W_2 = \{0, 1, 2\}$ . Let us start the chain from a projection  $\psi_0 = \psi_{m_1[\{0\}, \{1\}]}m_2[\{0\}, \{2\}]$ . This projection allows only for one pattern  $\langle [0, 1]; [0, 2] \rangle$ , i.e., the concept lattice is easily found. Then we increase the complexity of a projection by allowing more patterns. For example, we can enrich the first component of a tuple without affecting the second one, i.e., a projection  $\psi_1 = \psi_{m_1[\{0,1\}, \{0,1\}]}m_2[\{0\}, \{2\}]$ . This projection allows for 3 patterns, i.e., any possible interval of the first component and only one interval  $[0, 2]$  for the second component. Let us notice that it is not hard to find preimages for  $\psi_0$  in  $\psi_1(D)$ . Indeed, for any pattern  $p$  from  $\psi_0(D)$  one should just modify either the left side of the first interval of  $p$  by one value, or the right side of the first interval of  $p$ .*

*Then we can introduce a projection that slightly enrich the second component of a tuple, e.g.,  $\psi_2 = \psi_{m_1[\{0,1\}, \{0,1\}]}m_2[\{0,1\}, \{1,2\}]$  and finally we have  $\psi_3 = \psi_{m_1[W_1, W_1]}m_2[W_2, W_2]$ . Finding preimages in this chain is not a hard problem, since on every set we can only slightly change left and/or right side of the second interval in a tuple. Thus, starting from a simple projection and making transitions from one projection to another, we can cut unpromising branches and efficiently find the set of interesting patterns.*

### 3.3 Algorithms

	<p><b>Data:</b> A pattern structure <math>\mathbb{P}</math>, a chain of projections <math>\Psi = \{\psi_0, \psi_1, \dots, \psi_k\}</math>, a measure <math>\mathcal{M}</math> monotonic for the chain <math>\Psi</math>, and a threshold <math>\theta</math> for <math>\mathcal{M}</math>.</p>
1	<p><b>Function</b> <code>ExtendProjection(<math>i, \theta, \mathcal{P}_{i-1}</math>)</code></p>
	<p><b>Data:</b> <math>i</math> is the projection number to which we should extend (<math>0 &lt; i \leq k</math>), <math>\theta</math> is a threshold value for <math>\mathcal{M}</math>, and <math>\mathcal{P}_{i-1}</math> is the set of patterns for the projection <math>\psi_{i-1}</math>.</p>
	<p><b>Result:</b> The set <math>\mathcal{P}_i</math> of all patterns with the value of measure <math>\mathcal{M}</math> higher than the threshold <math>\theta</math> for <math>\psi_i</math>.</p>
2	<p><math>\mathcal{P}_i \leftarrow \emptyset;</math></p>
3	<p><i>/* Put all preimages in <math>\psi_i(\mathbb{P})</math> for any pattern <math>p</math> */</i></p>
4	<p><b>foreach</b> <math>p \in \mathcal{P}_{i-1}</math> <b>do</b></p>
5	<p>    <math>\mathcal{P}_i \leftarrow \mathcal{P}_i \cup \text{Preimages}(i, p)</math></p>
6	<p><i>/* Filter patterns in <math>\mathcal{P}_i</math> to have a value of <math>\mathcal{M}</math> higher than <math>\theta</math> */</i></p>
7	<p><b>foreach</b> <math>p \in \mathcal{P}_i</math> <b>do</b></p>
8	<p>    <b>if</b> <math>\mathcal{M}_{\psi_i}(p) \leq \theta</math> <b>then</b></p>
9	<p>      <math>\mathcal{P}_i \leftarrow \mathcal{P}_i \setminus \{p\}</math></p>
10	<p><b>Function</b> <code>Algorithm_θ-Σοφια</code></p>
	<p><b>Result:</b> The set <math>\mathcal{P}</math> of all patterns with a value of <math>\mathcal{M}</math> higher than the threshold <math>\theta</math> for <math>\mathbb{P}</math>.</p>
11	<p><i>/* Find all patterns in <math>\psi_0(\mathbb{P})</math> with a value of <math>\mathcal{M}</math> higher than <math>\theta</math> */</i></p>
12	<p><math>\mathcal{P} \leftarrow \text{FindPatterns}(\theta, \psi_0);</math></p>
13	<p><i>/* Run through out the chain <math>\Psi</math> and find the result patterns */</i></p>
14	<p><b>foreach</b> <math>0 &lt; i \leq k</math> <b>do</b></p>
15	<p>    <math>\mathcal{P} \leftarrow \text{ExtendProjection}(i, \theta, \mathcal{P});</math></p>

**Algorithm 1:** The  $\vartheta$ - $\Sigma\omicron\phi\alpha$  algorithm for finding patterns in  $\mathbb{P}$  with a value of a measure  $\mathcal{M}$  higher than a threshold  $\theta$ .

Given a measure monotonic w.r.t. a chain of projections, if we are able to find all preimages of any element in the fixed set of  $\psi_i$  that belong to a fixed set of  $\psi_{i+1}$ , then we can find all patterns of  $\mathbb{P}$  with a value of  $\mathcal{M}$  higher than a given threshold  $\theta$ . We call this algorithm  $\vartheta$ - $\Sigma\omicron\phi\alpha$  and its pseudocode is given in Algorithm 1. In lines 11-12 we find all patterns for  $\psi_0(\mathbb{P})$  satisfying the constraint that a value of  $\mathcal{M}$  is higher than a threshold. Then in lines 13-15 we iteratively extend projections from simpler to more detailed ones. The extension is done by constructing the set  $\mathcal{P}_i$  of preimages of the set  $\mathcal{P}_{i-1}$  (lines 2-5) and then by removing the patterns that do not satisfy the constraint from  $\mathcal{P}_i$  (lines 6-9).

The algorithm is sound and complete, since first, a pattern  $p$  is included

into the set of preimages of  $p$  ( $\psi(p) = p$ ) and second, if we remove a pattern  $p$  from the set  $\mathcal{P}$ , then the value  $\mathcal{M}(p) < \theta$  and, hence, the measure value of any preimage of  $p$  is less than  $\theta$  by the projection chain monotonicity of  $\mathcal{M}$ . The worst case time complexity of  $\vartheta$ - $\Sigma\circ\varphi\alpha$  algorithm is

$$\begin{aligned} \mathbb{T}(\vartheta\text{-}\Sigma\circ\varphi\alpha) &= \mathbb{T}(\text{FindPatterns}(\psi_0)) + \\ &+ k \cdot \max_{0 < i \leq k} |\mathcal{P}_i| \cdot (\mathbb{T}(\text{Preimages}) + \mathbb{T}(\mathcal{M})), \end{aligned} \quad (4)$$

where  $k$  is the number of projections in the chain,  $\mathbb{T}(\mathcal{X})$  is time for computing operation  $\mathcal{X}$ . Since projection  $\psi_0$  can be chosen to be very simple, in a typical case the complexity of  $\text{FindPatterns}(\theta, \psi_0)$  can be low or even constant. The complexities of  $\text{Preimages}$  and  $\mathcal{M}$  depend on the measure, the chain of projections, and the kind of patterns. In many cases  $\max_{0 < i \leq k} |\mathcal{P}_i|$  can be exponential in the size of the input, because the number of patterns can be exponential. It can be a difficult task to define the threshold  $\theta$  such that the maximal cardinality of  $\mathcal{P}_i$  is not larger than a given number. Thus, we introduce  $\Sigma\circ\varphi\alpha$  algorithm, which automatically adjusts threshold  $\theta$  ensuring that  $\max_{0 < i \leq k} |\mathcal{P}_i| < L$ . Here  $L$  can be considered as a constraint on the memory used by the algorithm.  $\Sigma\circ\varphi\alpha$  takes a pattern structure  $\mathbb{P}$ , a chain of projections  $\Psi = \{\psi_0, \psi_1, \dots, \psi_k\}$ , a measure  $\mathcal{M}$  monotonic for the chain  $\Psi$ , and a threshold  $L$  for the maximal number of preserved patterns. As the result it returns the threshold  $\theta$  ensuring that the cardinality of the set  $\mathcal{P}$  is bounded by  $L$  in any step of the algorithm and the set  $\mathcal{P}$  of all patterns with the value of measure  $\mathcal{M}$  higher than the threshold  $\theta$ . The only difference of  $\Sigma\circ\varphi\alpha$  algorithm w.r.t.  $\vartheta$ - $\Sigma\circ\varphi\alpha$  is that after performing an operation that changes the set  $\mathcal{P}$  (lines 12 and 15 of the Algorithm 1) it adjusts  $\theta$  in such a way that the cardinality of  $\mathcal{P}$  does not exceed the parameter  $L$ . It can be seen from (4) that  $\Sigma\circ\varphi\alpha$  has polynomial time complexity if  $\mathcal{M}$  and  $\text{Preimages}$  are polynomial. In particular, for interval pattern structures  $\Sigma\circ\varphi\alpha$  has polynomial complexity modulo the measure computation operation.

### 3.4 $\Sigma\circ\varphi\alpha$ Algorithm for Interval Tuple Data

In this subsection we consider a pattern structure  $\mathbb{K} = (G, (D_I, \sqcap), \delta)$ , where  $D_I$  is a semilattice of interval tuple descriptions. We say that every component of a tuple  $p$  corresponds to an attribute  $m \in M$ , the size of any tuple in  $D_I$  is  $|M|$ , and for any attribute  $m \in M$  we can denote the corresponding interval by  $m(p)$ . We also denote the value set of  $m$  by  $W_m$ . Since the set  $W_m$  is totally ordered we also denote by  $W_m^{(j)}$  and  $W_m^{(-j)}$  the sets containing  $j$  smallest and largest elements from  $W_m$ , respectively.

A projection chain for interval tuple data is formed in the same way as discussed in Example 5. We start from the projection containing only one pattern corresponding to the largest interval in each component, i.e., for an attribute  $m$  the projection is of the form  $\psi_m[W_m^{(1)}, W_m^{(-1)}]$ . Then to pass to a next projection, we select the attribute  $m$ , and for this attribute we extend the projection from  $\psi_m[W_m^{(j)}, W_m^{(-j)}]$  to  $\psi_m[W_m^{(j+1)}, W_m^{(-j-1)}]$ . Thus, there are

$k = \max_{m \in M} |W_m| \cdot |M|$  projections.

Finding preimages in this case is not hard, since to make a projection more detailed one should just extend the corresponding interval in left and/or on right end of the interval, i.e., there are only 4 possible preimages for a pattern when passing from one projection no another in this chain. Thus, we have proved the following

**Proposition 1** *The worst case complexity for  $\vartheta$ - $\Sigma\phi\alpha$  algorithm for interval tuple data is*

$$\mathbb{T}(\vartheta\text{-}\Sigma\phi\alpha_{\text{intervals}}) = \max_{m \in M} |W_m| \cdot |M| \cdot \max_{0 < i \leq k} |\mathcal{P}_i| \cdot \mathbb{T}(\mathcal{M}). \quad (5)$$

In particular, the complexity of  $\Sigma\phi\alpha$  for interval data is  $\max_{m \in M} |W_m| \cdot |M| \cdot L \cdot \mathbb{T}(\mathcal{M})$ , i.e., it is polynomial modulo complexity of the measure.

### 3.5 $\Sigma\phi\alpha$ Algorithm for Closed Patterns

Closed frequent itemsets are widely used as a condensed representation of all frequent itemsets since [13]. Here we show how we can adapt the algorithm for closed patterns. A closed pattern in  $\psi_{i-1}(\mathbb{P})$  is not necessarily closed in  $\psi_i(\mathbb{P})$ . However, the extents of  $\psi(\mathbb{P})$  are extents of  $\mathbb{P}$  [5]. Thus, we associate the closed patterns with extents and then work with extents instead of patterns, i.e., a pattern structure  $\mathbb{P} = (G, (D, \sqcap), \delta)$  is transformed into  $\mathbb{P}_C = (G, (D_C, \sqcap_C), \delta_C)$ , where  $D_C = 2^G$ . Moreover, for all  $x, y \in D_C$  we have  $x \sqcap_C y = (x^\diamond \sqcap y^\diamond)^\diamond$ , where diamond operator is computed in  $\mathbb{P}$  and  $\delta_C(g \in G) = \{g\}$ . Hence, every pattern  $p$  in  $D_C$  corresponds to a closed pattern  $p^\diamond$  in  $D$ . A projection  $\psi$  of  $\mathbb{P}$  induces a projection  $\psi_C$  of  $\mathbb{P}_C$ , given by  $\psi_C(X \subseteq G) = \psi(X^\diamond)^\diamond$  with  $(\cdot)^\diamond$  for  $\mathbb{P}$ .

### 3.6 $\Delta$ -measure and $\Sigma\phi\alpha$ Algorithm

In this subsection we show that  $\Delta$  is monotonic for any projection; it is a stronger condition than the one required by Definition 3.  $\Delta$ -measure works for closed patterns, and, hence, we identify every description by its extent (Subsection 3.5).

**Proposition 2**  *$\Delta$  is monotonic for any projection  $\psi$ .*

*Proof.* By properties of a projection, an extent of  $\psi(\mathbb{P})$  is an extent of  $\mathbb{P}$  [5]. Let us consider an extent  $E$  and an extent of its descendant in  $\psi(\mathbb{P})$ . Let us suppose that  $E_p$  is a preimage of  $E$  for the projection  $\psi$ . Since  $E_c$  and  $E_p$  are extents in  $\mathbb{P}$ , the set  $E_{cp} = E_c \cap E_p$  is an extent in  $\mathbb{P}$  (the intersection of two closed sets is a closed set). Since  $E_p$  is a preimage of  $E$ , then  $E_p \not\subseteq E_c$  (otherwise,  $E_p$  is a preimage of  $E_c$  and not of  $E$ ). Then,  $E_{cp} \neq E_p$  and  $E_{cp} \leq E_p$ . Hence,  $\Delta(E_p) \leq |E_p \setminus E_{cp}| \leq |E \setminus E_c|$ . So, given a preimage  $E_p$  of  $E$ ,  $(\forall E_c \subseteq E) \Delta(E_p) \leq |E \setminus E_c|$ , i.e.,  $\Delta(E_p) \leq \Delta(E)$ . Thus, we can use  $\Delta$ -measure in combination with  $\Sigma\phi\alpha$ .

Table 1: Patterns found for every projection in a chain for the example in Figure 1. Patterns are grey if they are removed for the corresponding projection and they are labeled with “-” if they have not yet been found.

#	Pattern Ext.	$\Delta$ -measure			
		$\psi_0$	$\psi_1$	$\psi_2$	$\psi_3$
1	$\{g_1, g_2, g_3, g_4, g_5, g_6\}$	6	2	2	2
2	$\{g_1, g_2, g_3, g_4\}$	-	4	1	1
3	$\{g_5, g_6\}$	-	2	2	2
4	$\{g_1\}$	-	-	1	1
5	$\{g_2, g_3, g_4\}$	-	-	3	2
6	$\{g_4\}$	-	-	-	1

### 3.7 Example of $\Delta$ -Stable Patterns in Interval Tuple Data

Let us consider the example in Figure 1 and show how we can find all  $\Delta$ -stable patterns with a threshold  $\theta = 2$ . The chain of projections for this example is given in Example 5, it contains 4 projections:

$$\begin{aligned} \psi_0 &= \psi_{m_1[\{0\},\{1\}]m_2[\{0\},\{2\}]} & \psi_1 &= \psi_{m_1[\{0,1\},\{0,1\}]m_2[\{0\},\{2\}]} \\ \psi_2 &= \psi_{m_1[\{0,1\},\{0,1\}]m_2[\{0,1\},\{1,2\}]} & \psi_3 &= \psi_{m_1[\{0,1\},\{0,1\}]m_2[\{0,1,2\},\{0,1,2\}]} \end{aligned}$$

Since we are looking for closed patterns, every pattern can be identified by its extent. In Table 1 all patterns are given by their extents, i.e., by elements of  $D_C$ . For every pattern  $\Delta$ -measure is shown for every  $\psi_i$ . A cell is shown in grey if the pattern is no more considered (the value of  $\Delta$  less than 2). A cell has a dash “-”, if a pattern in the row has not been generated for this projection.

For the example in Figure 1 the global process is as follows. At the beginning  $\psi_0(D_I)$  contains only one element corresponding to pattern extent 123456 (a short cut for  $\{g_1, g_2, g_3, g_4, g_5, g_6\}$ ) with a description  $\langle [0, 1]; [0, 2] \rangle$ . Then, in  $\psi_1(G, (D_I, \Pi), \delta)$  possible preimages of 123456 are patterns with descriptions  $\langle 0; [0, 2] \rangle$  and  $\langle 1; [0, 2] \rangle$  given by pattern extents 1234 and 56, respectively. Then we continue with these three patterns which are all  $\Delta$ -stable for the moment. The pattern extents 123456 and 56 have no preimages for the transition  $\psi_1 \rightarrow \psi_2$ , while the pattern extent 1234 has two preimages with descriptions  $\langle 0; [0, 1] \rangle$  and  $\langle 0; [1, 2] \rangle$  for this projection, which correspond to pattern extents 1 and 234. The first one is not  $\Delta$ -stable and thus is no more considered. Moreover, the pattern extent 1234 is not  $\Delta$ -stable (because of 234) and should also be removed. Finally, in transition  $\psi_2 \rightarrow \psi_3$  only extent-pattern 234 has a preimage, a pattern extent 4, which is not  $\Delta$ -stable. In such a way, we have started from a very simple projection  $\psi_0$  and achieved the projection  $\psi_3$  that gives us the  $\Delta$ -stable patterns of the target pattern structure.

## 4 Experiments and Discussion

In this section we compare our approach to approaches based on postfiltering. Indeed, there is no approach that can directly mine stable-like pattern, e.g. stable,  $\Delta$ -stable or robust patterns. The known approaches use postfiltering to mine such kind of patterns [14, 12, 2, 16]. Recently it was also shown that it is more efficient to mine interval tuple data without binarization [8]. In their paper the authors introduce algorithm `MinIntChange` for working directly with interval tuple data. Thus we compare `Σοφια` and `MinIntChange` for finding  $\Delta$ -stable patterns. We find  $\Delta$ -stable concepts by `Σοφια` and then adjust frequency threshold  $\theta$  such that all  $\Delta$ -stable patterns are among the frequent ones.

The experiments are carried out on an “Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz” computer with 8Gb of memory under Ubuntu 14.04 operating system. The algorithms are not parallelized and are coded in C++.

### 4.1 Dataset Simplification

For interval tuple data stable patterns can be very deep in the search space, such that neither of the algorithms can find them quickly. Thus, we join some similar values for every attribute in an interval in the following way. Given a threshold  $0 < \beta$ , two consequent numbers  $w_i$  and  $w_{i+1}$  from a value set  $W$  are joint in the same interval if  $w_{i+1} - w_i < \beta$ . In order to properly set the threshold  $\beta$ , we use another threshold  $0 < \gamma < 1$ , which is much easier to set.

If we assume that the values of the attribute  $m$  are distributed around several states with centers  $\tilde{w}^1, \dots, \tilde{w}^l$ , then it is natural to think that the difference between the closest centers  $\text{abs}(\tilde{w}^i - \tilde{w}^{i\pm 1})$  are much larger than the difference between the closest values. Ordering all values in the increasing order and finding the maximal difference  $\delta_{\max}$  can give us an idea of typical distance between the states in the data. Thus,  $\gamma$  is defined as a proportion of this distance that should be considered as a distance between states, i.e., we put  $\beta = \gamma \cdot \delta_{\max}$ . If the distance between closest values in  $W$  are always the same, then even  $\gamma = 0.99$  does not join values in intervals. However, if there are two states and the values are distributed very closely to one of these two states, then even  $\gamma = 0.01$  can join values into one of two intervals corresponding to the states.

### 4.2 Datasets

We take several datasets from the Bilkent University database <sup>1</sup>. The datasets are summarized in Table 2. The names of datasets are given by standard abbreviations used in the database of Bilkent University. For every dataset we provide the number of objects and attributes and the threshold  $\gamma$  for which the experiments are carried out. For example, database `EM` has 61 objects, 9 numeric attributes, and the threshold  $\gamma$  is set to 0.3. Categorical attributes and rows with missing values, if any, are removed from the datasets.

<sup>1</sup><http://funapp.cs.bilkent.edu.tr/DataSets/>

Table 2: Runtime in seconds of  $\Sigma\phi\alpha$  and `MinIntChange` for different datasets.

DS	# Objs	# Attrs	$\gamma$	$\Delta$	# Ptrns	$\theta$	$t_{\Sigma\phi\alpha}$	$t_{MIC}$
EM	61	9	0.3	3	3	21	< 0.1	57
BK	96	4	0.3	4	50	46	< 0.1	11
CN	105	20	0.8	2	5362	30	2.4	28
CU	108	5	0.3	5	4	27	< 0.1	1.5
FF	125	3	0.3	6	3	48	< 0.1	1
AP	135	4	0.01	5	1	19	< 0.1	34
EL	211	12	0.3	6	33	83	< 0.1	34
BA	337	16	0.5	4	736	91	1.5	32
AU	398	7	0.3	7	17	234	0.7	73
HO	506	13	0.8	10	1	340	0.7	57
QU	2178	25	0.3	40	1	659	1.3	28
AB	4177	8	0.3	46	3	1400	11	86
CA	8192	21	0.3	85	6	2568	112	24
PT	9065	48	0.3	2	1	2	45	14

### 4.3 Experiments

In Table 2 we show the computation time for finding the best  $\Delta$ -stable pattern (or patterns if they have the same value for  $\Delta$ -measure) for  $\Sigma\phi\alpha$  and for `MinIntChange`. The last algorithm is abbreviated as MIC. Since `MinIntChange` algorithm sometimes produces too many patterns, i.e., we do not have enough memory in our computer to check all of them, we interrupt the procedure and show the corresponding time in grey. We also show the number of the best patterns and the corresponding threshold  $\Delta$ . The support threshold  $\theta$  for finding the best  $\Delta$ -stable patterns is also shown. For example, dataset CN contains 5362 best  $\Delta$ -stable patterns, all having a  $\Delta$  of 2. To find all these patterns with a postfiltering, we should mine frequent patterns with a support threshold lower than 30 or  $\frac{30}{105} = 30\%$ .  $\Sigma\phi\alpha$  computes all these patterns in 2.4 seconds, while `MinIntChange` requires at least 28 seconds and the procedure was interrupted without continuation.

As we can see,  $\Sigma\phi\alpha$  is significantly faster than `MinIntChange` in all datasets. In the two datasets CA and PT, `MinIntChange` was stopped before computing all patterns and the runtime did not exceed the runtime of  $\Sigma\phi\alpha$ . However, in both cases, `MinIntChange` achieved less than 10% of the required operations.

## 5 Conclusion

In this paper we have introduced a new class of interestingness measures that are monotonic w.r.t. a chain of projections. We have designed a new algorithm, called  $\Sigma\phi\alpha$ , which is able to find the best patterns w.r.t. such interestingness measures in polynomial time for interval tuple data. The experiments reported in the paper are the witness of the efficiency of the  $\Sigma\phi\alpha$  algorithms compared to indirect approaches based on postfiltering. Many future research directions are possible. Different measures should be studied in combination with  $\Sigma\phi\alpha$ .



One of them is robustness, which is very close to stability and can be applied to nonbinary data. Moreover, the choice of a projection chain is not a simple one and can affect the algorithm efficiency. Thus, a deep study of suitable projection chains should be carried out.

**Acknowledgments:** this research was supported by the Basic Research Program at the National Research University Higher School of Economics (Moscow, Russia) and by the BioIntelligence project (France).

## References

- [1] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994.
- [2] Aleksey Buzmakov, Elias Egho, Nicolas Jay, Sergei O. Kuznetsov, Amedeo Napoli, and Chedy Raïssi. On Projections of Sequential Pattern Structures (with an application on care trajectories). In *Proc. 10th Int. Conf. Concept Lattices Their Appl.*, pages 199—208, La Rochelle, France, 2013.
- [3] Aleksey Buzmakov, Sergei O. Kuznetsov, and Amedeo Napoli. Scalable Estimates of Concept Stability. In Christian Sacarea, Cynthia Vera Glodeanu, and Mehdi Kaytoue, editors, *Form. Concept Anal.*, volume 8478 of *Lecture Notes in Computer Science*, pages 161–176. Springer Berlin Heidelberg, 2014.
- [4] Jie Cao, Zhiang Wu, and Junjie Wu. Scaling up cosine interesting pattern discovery: A depth-first method. *Inf. Sci. (Ny)*, 266(0):31–46, 2014.
- [5] Bernhard Ganter and Sergei O. Kuznetsov. Pattern Structures and Their Projections. In Harry S. Delugach and Gerd Stumme, editors, *Concept. Struct. Broadening Base*, volume 2120 of *Lecture Notes in Computer Science*, pages 129–142. Springer Berlin Heidelberg, 2001.
- [6] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, 1st edition, 1999.
- [7] Jiawei Han, Jianyong Wang, Ying Lu, and P Tzvetkov. Mining top-k frequent closed patterns without minimum support. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE Int. Conf.*, pages 211–218, 2002.
- [8] Mehdi Kaytoue, Sergei O. Kuznetsov, and Amedeo Napoli. Revisiting Numerical Pattern Mining with Formal Concept Analysis. In *IJCAI 2011, Proc. 22nd Int. Jt. Conf. Artif. Intell. Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 1342–1347, 2011.
- [9] Sergei O. Kuznetsov. On stability of a formal concept. *Ann. Math. Artif. Intell.*, 49(1-4):101–115, 2007.

- [10] Sergei O. Kuznetsov and Mikhail V. Samokhin. Learning Closed Sets of Labeled Graphs for Chemical Applications. In Stefan Kramer and Bernhard Pfahringer, editors, *Inductive Log. Program. SE - 12*, volume 3625 of *Lecture Notes in Computer Science*, pages 190–208. Springer Berlin Heidelberg, lecture no edition, 2005.
- [11] Heikki Mannila, Hannu Toivonen, and A Inkeri Verkamo. Efficient Algorithms for Discovering Association Rules. In *Knowl. Discov. Data Min.*, pages 181–192, 1994.
- [12] Fabian Moerchen, Michael Thies, and Alfred Ultsch. Efficient mining of all margin-closed itemsets with applications in temporal knowledge discovery and classification by compression. *Knowl. Inf. Syst.*, 29(1):55–80, 2011.
- [13] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Efficient Mining of Association Rules Using Closed Itemset Lattices. *Inf. Syst.*, 24(1):25–46, 1999.
- [14] Camille Roth, Sergei A. Obiedkov, and Derrick G. Kourie. On succinct representation of knowledge community taxonomies with formal concept analysis. *Int. J. Found. Comput. Sci.*, 19(02):383–404, April 2008.
- [15] Eirini Spyropoulou, Tijn De Bie, and Mario Boley. Interesting pattern mining in multi-relational data. *Data Min. Knowl. Discov.*, (April):1–42, 2013.
- [16] Nikolaj Tatti, Fabian Moerchen, and Toon Calders. Finding Robust Itemsets under Subsampling. *ACM Trans. Database Syst.*, 39(3):1–27, October 2014.
- [17] Jilles Vreeken and Nikolaj Tatti. Interesting Patterns. In Charu C Aggarwal and Jiawei Han, editors, *Freq. Pattern Min.*, pages 105–134. Springer International Publishing, 2014.
- [18] Geoffrey I. Webb. Self-sufficient itemsets. *ACM Trans. Knowl. Discov. Data*, 4(1):1–20, January 2010.
- [19] Geoffrey I. Webb. Filtered-top-k association discovery. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, 1(3):183–192, 2011.
- [20] Dong Xin, Hong Cheng, Xifeng Yan, and Jiawei Han. Extracting redundancy-aware top-k patterns. In *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discov. data Min. - KDD '06*, page 444, New York, New York, USA, August 2006. ACM Press.
- [21] Xifeng Yan, Hong Cheng, Jiawei Han, and Philip S. Yu. Mining significant graph patterns by leap search. In *Proc. 2008 ACM SIGMOD Int. Conf. Manag. data - SIGMOD '08*, pages 433–444, New York, New York, USA, June 2008. ACM Press.

- [22] Xifeng Yan, Jiawei Han, and Ramin Afshar. CloSpan: Mining Closed Sequential Patterns in Large Databases. In *Proc. SIAM Int'l Conf. Data Min.*, pages 166–177, 2003.
- [23] Hong Yao and Howard J Hamilton. Mining itemset utilities from transaction databases. *Data Knowl. Eng.*, 59(3):603–626, 2006.