Constraint-programming Approach for Multiset and Sequence Mining

Pablo Gay, Beatriz López and Joaquim Meléndez
University of Girona, Campus Montilivi, P4 Building, E17071, Girona, Spain

Keywords: Data Mining, Itemset Mining, Multiset Mining, Sequence Mining, Pattern Discovery.

Abstract: Constraint-based data mining is a field that recently has started to receive more attention. Describing a problem through a declarative model enables very descriptive and easy to extend implementations. Our work uses a previous itemset mining model in order to extend it with the capabilities to discover different and interesting patterns that have not been explored yet: multisets and sequences. The classic example domain is the retailer organizations, trying to mine the most common combinations of items bought together. Multisets would allow mining not only this itemsets but also the quantities of each item and sequences the order in with the items are retrieved. In this paper, we provide the background of the original work and we describe the modifications done to the model to extend it and support these new patterns. We also test the new models using real world data to prove their feasibility.

1 INTRODUCTION

Itemset mining, a problem originated in the study of retailer organization databases, has been a major concern in the machine learning community due its impact on different areas, such as games, census, traffic accidents, among other (Rep, 2012). The learning objective is to find the most common combination of products bought toghether. After itemset mining, sequence (Agrawal and Srikant, 1995) and multiset (David and Nourine, ence) mining algorithms have been developed. The importance of sequences is shown in fields as recommendations (Burke, 1999), web browsing (Cadez et al., 2000), health care (Zhang et al., 2003) or music (Brand, 1998) for instance. In sequence mining, item order matters, while in multisets item repetitions is taken into account.

The original itemset mining algorithm known is Apriori (Agrawal and Srikant, 1994) and consists in three phases. First, it builds an item lattice. Second it scans the lattice looking for itemsets that have at least a support of a user-specified threshold. And third, and in order to build associational rules, builds up the appropriate relationship between premises and consequences. A similar process follows with sequences. Since the approach is bread first search, it is computationally expensive, and other authors have been explored several alternatives to improve its efficiency. (Srikant and Agrawal, 1996) proposed the use of taxonomies, some primitive data structures to count the support of patterns, sliding windows to relax the definition of frequent sequences, and time constraints to discriminate whenever two elements of a transaction belong to the same sequence or not. (David and Nourine, ence) goes one step further by mixing sequences and multisets. All of these approaches are procedural based, and they search in one way or another a tree structure.

Recently, there has been an interest to use declarative approaches for data mining (Bonchi and Lucchese, 2007). In the particular case of itemsets, (De Raedt et al., 2008) proposes to use constraint programming. Although constraint programming is a well-studied field in computer science, its use for itemset mining had gone unnoticed until then. The main achievement is the reformulation of the itemset mining problem as a constraint satisfaction problem, exploiting current and powerful solvers to find out the frequent itemset patterns. The framework proposed at (De Raedt et al., 2008) is revisited in (De Raedt et al., 2010), where the authors explore in deep the formulation, propose models for frequent, closed, and weighted itemset mining, and analyze the efficiency gained with solvers. The constraint programming approach enables the extension, in a declarative manner, of the data mining problem by the addition of new constraints.

Our work concerns the extension of the previous work of (De Raedt et al., 2010), to mine two different kind of frequent patterns: multisets and sequences.

Gay P., López B. and Meléndez J..
Constraint-programming Approach for Multiset and Sequence Mining.
DOI: 10.5220/0004135302120220
In *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval* (KDIR-2012), pages 212-220
ISBN: 978-989-8565-29-7

Our ultimate goal is to provide a constraint programming framework for frequent multiset and sequence mining.

This paper is organized as follows. First, in Section 2 we contextualize our research within some other related work. Next, we provide in Section 3 the basis of the constraint programming approach in (De Raedt et al., 2010), that is then extended with our new proposals in Sections 4 (multiset mining) and 5 (sequence mining). The experimental results and their discussion is presented in Section 6. We end with some conclusions and future work directions in Section 7.

2 RELATED WORK

Constraint-based mining has been understood in frequent itemset mining as the process of defining constraints in the mining process. That means, that we find itemset patterns by constraining their length, duration, gaps between adjacent items, prefixes, and so on (Han et al., 2007). These methods follow a procedural approach to find out the frequent patterns. Our work concerns on the use of the constraint programming paradigm for mining frequent patterns, following a declarative definition of the mining problem, and letting the solvers find the patterns, as in (De Raedt et al., 2010).

An example of such systems is Molfea (De Raedt and Kramer, 2001), which mines chemical structures for sequences of atoms and bonds that are defined by some user criteria. These criteria are primitives that represent fragments of the sequences of atoms and may require having a minimum frequency on the database. Another example is ConQuest (Bonchi et al., 2009), a constraint-based querying system, which interacts with the user to achieve the desired learning results. MusicDFS (Soulet et al., 2006) is another application example of a tool which implements its own efficient deep search first algorithm to mine constrained patterns. These kind of frameworks have in common that they focus on the use of constraints in the search process to improve mining results whereas our approach focuses on create a model definition to solve the problem of mining multisets and sequence patterns using a constraint programming approach.

Other related works are in the field of plan recognition. The plan recognition problem (Schmidt et al., 1978) takes as input a set of sequential actions performed by some actors within the system and tries to organize these actions in order to recreate the plan and to infer the pursued goal. Moreover, the plans can be considered sequential patterns if they are mined using data from several actors. Recently, this community has also moved to consider constraint-satisfaction formulations to define their problem (Gal et al., 2012).

3 BACKGROUND

In this work, the modelization of frequent itemset problem has been considered as starting point to extend it towards multiset and ordered itemset mining. However, formalization can be generalized to cover closed, maximal or typologies of itemset mining problems.

As it has been commented above, our model is based on the original and simpler design of (Guns et al., 2011). It is a constraint programming approach for itemset mining. We explain the essentials of the method (the problem definition and the constraint programming model) in this section to proceed afterward with the definition of our models.

3.1 Problem Statement

The original motivation of itemset mining was on the retailers, from which most of the terminology has been derived. The input information is a database containing all the products acquired by the clients according to transactions. The output, the most frequent items bought.

Let $I = \{1,\ldots,m\}$ be a subset of $\mathbb{N}$ that map some items as for example $\{A,B,\ldots,M\}$ and $\mathcal{A} = \{1,\ldots,n\}$ a set of transaction identifiers. Now, let $\mathcal{D}'$ be a set of tuples with a transaction id t and an itemset I. Finally, let $\mathcal{D}$ be the binary representation of $\mathcal{D}'$ such that:

$$\mathcal{D}' = \{(t,I) \mid t \in \mathcal{A}, I \subseteq I, \forall i \in I : \mathcal{D}_{t,i} = 1\} \quad (1)$$

where $\mathcal{D}_{t,i}$ is the position (t,i) of the $\mathcal{D}$ matrix. Using this definition itemsets can be discriminated using their id and $\mathcal{D}$ contains 1s only when the itemset contains that specific item. Table 1 shows an example of this representations.

The occurrences of a specific itemset I within the matrix $\mathcal{D}$, that is, the coverage φ, is defined as follows:

$$\varphi_{\mathcal{D}}(I) = \{t \in \mathcal{A} \mid \forall i \in I : \mathcal{D}_{t,i} = 1\} \quad (2)$$

For example, using matrix $\mathcal{D}$ from Table 1, $\varphi_{\mathcal{D}}(\{2,3\}) = \{5,8,9,10\}$.

The amount of transactions that contain the itemset is denoted as *support* and is defined as:

$$support_{\mathcal{D}}(I) = |\varphi_{\mathcal{D}}(I)| \quad (3)$$

So using again the previous example, $support_{\mathcal{D}}(\{2,3\}) = |\varphi_{\mathcal{D}}(\{2,3\})| = |\{5,8,9,10\}| = 4$

Table 1: Example of item database. Left: original itemset database. Central ($\mathcal{D}'$): transactions database. Right ($\mathcal{D}$): binary representation.

Transactions	(id, Itemset)
$\{C\}$	(1, $\{3\}$)
$\{B\}$	(2, $\{2\}$)
$\{A,C\}$	(3, $\{1,3\}$)
$\{B,D\}$	(4, $\{2,4\}$)
$\{B,C\}$	(5, $\{2,3\}$)
$\{C,D\}$	(6, $\{3,4\}$)
$\{A,D\}$	(7, $\{1,4\}$)
$\{B,C,D\}$	(8, $\{2,3,4\}$)
$\{A,B,C\}$	(9, $\{1,2,3\}$)
$\{A,B,C,D\}$	(10, $\{1,2,3,4\}$)

$\mathcal{D}'$

id	A	B	C	D
1	0	0	1	0
2	0	1	0	0
3	1	0	1	0
4	0	1	0	1
5	0	1	1	0
6	0	0	1	1
7	1	0	0	1
8	0	1	1	1
9	1	1	1	0
10	1	1	1	1

$\mathcal{D}$

Frequent itemset mining consists on finding all the itemsets which support is equal or higher than a certain threshold θ. That is:

$$\text{frequent itemsets} = \{I \mid I \in \mathcal{I}, support_D(I) \geq \theta\} \quad (4)$$

3.2 Constraint Programming Model

First of all, (Guns et al., 2011) define a boolean variable I_i for each individual item. Itemsets I are then represented as a collection of this binary variables. Another set of binary variables, T_t, represent the transactions of the set of T that cover a given itemset, $T = \varphi_{\mathcal{D}}(I)$. For example, transaction 8 of Table 1 is represented by the following setting of the binary variables settings: $I_1 = 0, I_2 = 1, I_3 = 1, I_4 = 1, T_8 = 1$ for $\varphi_{\mathcal{D}}(2,3,4)$.

Thanks to this new binary variables, the coverage and support constraints are formulated. The coverage constraints is modeled as follows:

$$T = \varphi_{\mathcal{D}}(I) \Leftrightarrow (\forall t \in \mathcal{A} : T_t = 1 \leftrightarrow \sum_{i \in I} I_i(1 - \mathcal{D}_{t,i}) = 0) \quad (5)$$

The second one, the frequency constrain, requires computing if the sum of the binary vector T is greater or equal to the θ threshold, as follows:

$$|T| > \theta \Leftrightarrow \sum_{t \in \mathcal{A}} T_t \geq \theta \quad (6)$$

Therefore, the itemset mining from a constraint programming approach consist on finding the set

$$(I,T) | I \subseteq \mathcal{I}, T \subseteq A, T = \varphi_{\mathcal{D}}(I), |T| > \theta \quad (7)$$

4 MULTISET MINING

The previous section presented how to find frequent itemsets. This kind of patterns can be very useful, since a lot of databases can be transformed into the binary representation shown at Equation 1. In the classical scenario, the retailer database, each row will be considered as a client transaction and each column as an item. Then the resulting itemset patterns are the most common products bought together.

However, using this same scenario, we can see that we could extract more information. Frequent patterns at this point represent the products commonly purchased together, but the amount of each product is still unknown. We can know that beer and crisps are bought together, but probably there is a proportional relation among them, like for each six-pack of beer, two bags of crisps. Consequently, the original problem definition and the model should be extended to learn patterns with such property.

4.1 Problem Statement

Until now, we have been working under the assumption that we were using sets of items called itemsets that did not contain any item repeated. Now itemsets can contain the same item more than once. Hence, we need to modify also our description of $\mathcal{D}'$ and $\mathcal{D}$ to use multisets:

$$\mathcal{D}' = \{(t,I) \mid t \in \mathcal{A}, I \subseteq \mathcal{I}, \forall i \in I, \mathcal{D}_{t,i} = Q_i(I)\} \quad (8)$$

This new representation differs with the one at Equation 1 when assigning the value of $\mathcal{D}_{t,i}$. In this case instead of using a binary value to assert whenever the item exists or not in the itemset, we propose the function $Q_i(I)$, which return the cardinal of the set of all the items in I that are equal to i or, what is the same, the Quantity of i in I:

$$Q_i(I) = |\{k \in I \mid k = i\}| \quad (9)$$

Table 2 show an example of how this new matrix $\mathcal{D}$ is.

Table 2: Example of item database with repetition. Left ($\mathcal{D}'$): transaction database, where items are repeated. Right ($\mathcal{D}$): new representation using repetitions.

(T_{id}, Itemset)
(1, {2,3,2})
(2, {4})
(3, {1,3})
(4, {1})
(5, {2,3})
(6, {4})
(7, {3,4})
(8, {1,2,2,3})
(9, {1,2})
(10, {1,2,3})

$\mathcal{D}'$

T_{id}	A	B	C	D
	1	2	3	4
1	0	**2**	1	0
2	0	0	0	1
3	1	0	1	0
4	1	0	0	0
5	0	1	1	0
6	0	0	0	1
7	0	0	1	1
8	1	**2**	1	0
9	1	1	0	0
10	1	1	1	0

$\mathcal{D}$

Since the input matrix now instead of containing binary values it contains the specific amounts of items in the transactions, the coverage definition (Equation 2) must change accordingly:

$$\varphi^R_{\mathcal{D}}(I) = \{t \in \mathcal{A} \mid \forall i \in I, \mathcal{D}_{t,i} \geq Q_i(I)\} \quad (10)$$

With this new definition we assure that support is estimated properly, since itemsets with high amounts of repeated items than the desired one provide support but not vice versa. For example, $\{1,3,3\}$ supports the pattern $\{1,3\}$ but not the $\{1,3,3,3\}$ one. This new definition does not affect the way support is estimated (i.e. Equation 3), but we redefine it for convenience, since the coverage set to be taken into account is different:

$$support^R_{\mathcal{D}}(I) = |\varphi^R_{\mathcal{D}}(I)| \quad (11)$$

Therefore, the multiset mining problem from a constraint programming approach consists on finding the set

$$\text{frequent itemsets} = \{I \mid I \in I, support^R_{\mathcal{D}}(I) \geq \theta\} \quad (12)$$

Observe that the problem definition for multisets is still compatible with the original model since it is equivalent to discover patterns from multisets where at most there is one repetition for item.

4.2 Constraint Programming Model

According to the problem definition, the model main extension is related to the coverage constraint. Now, the instantiated vector T contain a 1 in the t_{th} only if the amount of each item i within a certain itemset I is greater or equal in the t_{th} transaction of the matrix $\mathcal{D}$:

$$\forall t \in \mathcal{A} : T_t = 1 \leftrightarrow \bigwedge_{i \in I} \mathcal{D}_{t,i} \geq Q_i(I) \quad (13)$$

Table 3: Output from the multiset mining.

A	B	C	D		
1	2	3	4	Support	Itemset
0	0	0	1	3	{D}
0	2	0	0	2	{B,B}
0	2	1	0	2	{B,B,C}
1	1	1	0	2	{A,B,C}
1	0	1	0	3	{A,C}
0	1	1	0	4	{B,C}
0	0	1	0	6	{C}
1	1	0	0	3	{A,B}
1	0	0	0	5	{A}
0	1	0	0	5	{B}

The support constraint and the problem formulation from the remain the same as in Section 3.2.

4.3 Example

Here we provide an illustrative example of the model proposed. We use example provided in Table 2. The θ threshold is 2 and it represents the 20% of the database.

Thus, the multiset $\{B,B,C\}$ has its representation in the $\mathcal{D}$ matrix as $\{0,2,1,0\}$. To estimate the coverage of $\{B,B,C\}$, all the transactions from $\mathcal{D}$ must be checked to see if they meet the specifications shown at Equation 10. Transactions 1 and 8 do, so $\varphi^R_{\mathcal{D}}(\{0,2,1,0\}) = \{1,8\}$. The next step is calculating the support, so as it is the cardinal of the coverage, $support^R_{\mathcal{D}}(\{0,2,1,0\}) = |\varphi^R_{\mathcal{D}}(\{0,2,1,0\})| = |\{1,8\}| = 2$. And finally, multiset $\{0,2,1,0\}$ will be considered a frequent patterns since this support is greater or equal to the previously specified θ.

Following the same procedure, multiset $\{C,D\}$ is represented as $\{0,0,1,1\}$ in $\mathcal{D}$. Then, $\varphi^R_{\mathcal{D}}(\{0,0,1,1\}) = \{7\}$ since 7 is the only transaction that has elements C and D greater or equal to 1. Finally, $support^R_{\mathcal{D}}(\{0,0,1,1\}) = |\varphi^R_{\mathcal{D}}(\{0,0,1,1\})| = |\{7\}| = 1$, which obviously is lower than the θ threshold an therefore it is not considered as a frequent pattern.

All the frequent multiset patterns found for the example are shown in Table 3.

5 SEQUENCE MINING

In this section we are considering itemsets with item order information but without any repetition (see Table 4).Back again to the retailers example, sequence mining allows knowing the order in which items are retrieved. Such knowledge can be useful for different optimization purposes, as the product distribution:

Table 4: Example of ordered itemsets database. Left ($\mathcal{D}'$): transaction database. Right ($\mathcal{D}$): representation including order.

(T_{id}, Itemset)
(1, {4})
(2, {3,2})
(3, {3,1,2})
(4, {1,3,2})
(5, {4,1})
(6, {3,4})
(7, {1,2,3})
(8, {3,4,2})
(9, {2,4,3,1})
(10, {1,4,3,2})

$\mathcal{D}'$

T_{id}	A	B	C	D
1	0	0	0	1
2	0	2	1	0
3	2	3	1	0
4	1	3	2	0
5	2	0	0	1
6	0	0	1	2
7	1	2	3	0
8	0	3	1	2
9	4	1	3	2
10	1	4	3	2

$\mathcal{D}$

the clients could find a simpler route to gather everything or the store manager could introduce in the mined routes products that usually are not bought by the clients but they could be interested in.

5.1 Problem Statement

In this case, itemsets are ordered sets of items, so in a certain itemset I, I_n would be its n_{th} item. Consequently, the matrix $\mathcal{D}$ must be able to represent order information. Our approach is similar to the ones presented before, keeping the definitions as simple as possible:

$$\mathcal{D}' = \{(t,I) \mid t \in \mathcal{A}, I \subseteq I, \forall i \in I, \mathcal{D}_{t,i} = O_i(I)\} \quad (14)$$

The main difference with Equation 1 is the binary value assigned to $\mathcal{D}_{t,i}$ to represent the existence of the item in the itemset. Here we have replaced it with the new function $O_i(I)$, where given a certain ordered itemset I and a certain item i returns the position of i within I, for example $O_4(\{2,4,3,1\}) = 2$. The *O*rder function, $O_i(I)$, is the following:

$$O_i(I) = k \mid \exists k \leq |I|, I_k = i \quad (15)$$

The coverage constraint needs to be redefined too. In the original work and in our previous extension, coverage is a measure that estimates how many times a combination of items or a quantity of items is contained by the database. The new coverage function we propose needs to change completely this paradigm. Instead of counting the items, the items must fulfill specific conditions between themselves. That means that if we want to count the support of a certain ordered itemset $\{C,A,B\}$, first we need to know which of the transactions in the database contain items where the item C goes before A and B, where item A goes after item C and before B and where item B goes after items C and A, or what's the same, which transactions in the database follow the order specified by the itemset. It is easy to see that these conditions can be also considered rules or constraints, what makes the constraint-based definition an excellent choice. The following equation shows how simple is to represent this conditions:

$$\varphi^O_{\mathcal{D}}(I) = \{t \in \mathcal{A} \mid \forall i,j \in I, i \neq j, (I_i < I_j) \rightarrow (\mathcal{D}_{t,i} < \mathcal{D}_{t,j})\} \quad (16)$$

The new coverage represents the set of all the transactions indices t where given all possible positions i and j (and $i \neq j$) within an itemset I, if item I_i appears before then item I_j then in the item in the i_{th} position of the transaction must appear also before the one at the j_{th}.

Consistently, the support is defined as follows:

$$support^O_{\mathcal{D}}(I) = |\varphi^O_{\mathcal{D}}(I)| \quad (17)$$

Finally, the sequence mining problem from a constraint programming approach consists on finding the set

$$\text{frequent itemsets} = \{I \mid I \in I, support_{\mathcal{D}^O}(I) \geq \theta\} \quad (18)$$

5.2 Constraint Programming Model

According to the problem definition, the coverage constraint form is modeled as a constrained programming approach as follows:

$$\forall t \in \mathcal{A} : T_t = 1 \leftrightarrow \bigwedge_{i \in I} \bigwedge_{j \in I \setminus \{i\}} (I_i < I_j) \rightarrow (\mathcal{D}_{t,i} < \mathcal{D}_{t,j}) \quad (19)$$

The model denoted by the above equation, comes straightforward from Equation 16, but it has three problems. The first and most important one is the following: the constraint checks the condition $(I_i < I_j) \rightarrow (\mathcal{D}_{t,i} < \mathcal{D}_{t,j})$ and if $\mathcal{D}_{t,i}$ is equals to 0 (i.e. the item does not belong to the transaction), a true assertion will be occur. Consequently, we add an additional condition to the constraint, as follows:

$$\forall t \in \mathcal{A} : T_t = 1 \leftrightarrow \bigwedge_{i \in I} \bigwedge_{j \in I \setminus \{i\}} (I_i < I_j) \rightarrow \ldots \\ \ldots \rightarrow ((\mathcal{D}_{t,i} > 0) \wedge (\mathcal{D}_{t,i} < \mathcal{D}_{t,j})) \quad (20)$$

The second problem relies in the constraint definition itself. It looks for how many transactions fulfill some denoted conditions between items, so the resulting patterns will have at minimum two items. In the case that patterns with single items were requested the procedure becomes the same one as the previous model: counting occurrences of items. In that case,

Table 5: Output from the sequence mining.

A	B	C	D		
1	2	3	4	Support	Itemset
1	2	0	0	4	{A,B}
1	0	2	0	3	{A,C}
1	3	2	0	2	{A,C,B}
0	1	2	0	2	{B,C}
2	0	1	0	2	{C,A}
0	2	1	0	5	{C,B}
0	0	1	2	2	{C,D}
2	0	0	1	2	{D,A}
0	2	0	1	2	{D,B}
0	0	2	1	2	{D,C}

the previous model should be used adding a new constraint limiting the pattern length like $|I| = 1$ can be used.

The third problem is that any solver fed with this model could return malformed patterns. If we look back to Table 4, the ordered itemset $\{0,2,1,0\}$ can be considered a pattern since half of the transactions contain this sequence. What makes inconsistent the model is that the solver could return $\{0,2,1,0\}$, $\{0,3,1,0\}$ or $\{0,3,2,0\}$ as well since we are checking the transactions precedences without any point of reference. All three of them have the same meaning: that the item C has a lower value of than item B so therefore it goes before, but only the first one should be returned. An auxiliary constrain like $\forall i \in I,\ i \leq |I|$ limiting the maximum value that the items can have should be more than enough to avoid this.

Finally, the support constraint and the problem formulation from the remain the same as in Section 3.2.

5.3 Example

To provide an illustrative example of our mode, we use the data shown in Table 4. We assume the threshold value: $\theta = 2$

Like in the previous example, the first thing to do is to translate the itemset to match the representation from Table 4. If we choose $\{C,B\}$ (transaction 2), then its representation is $\{0,2,1,0\}$. If we what know if it is a frequent pattern, from Equation 16, item C has a lower value (goes before) than item B, and all the transactions in matrix $\mathcal{D}$ that contain a lower value in item C than in item B support this itemset. That is, $\varphi^{O}_{\mathcal{D}}(\{0,2,1,0\}) = \{2,3,4,8,10\}$. Consequently, through Equation 17, $support^{O}_{\mathcal{D}}(\{0,2,1,0\}) = |\{2,3,4,8,10\}| = 5$ and since it is a value greater than the θ specified it is considered a frequent pattern.

All the frequent sequence patterns found for the example are shown in Table 5.

Table 6: Multiset patterns.

Support	Pattern
10%	2x frontpage, 2x tech, 1x msn-sports
12%	1x msn-sports
12%	1x frontpage, 1x news, 1x business
21%	1x news
21%	3x frontpage
22%	1x local
23%	1x tech
26%	1x on-air
32%	2x frontpage
51%	1x frontpage

6 EXPERIMENTATION

In order to test our approach with real data, the UCI (Frank and Asuncion, 2010) dataset corresponding to the MSNBC anonymous web navigation has been used.

6.1 Experimental Setup

The raw information provided by the dataset consist on 1000000 data lines, each one is the activity of a user session and contains a sequence of numbers from 1 to 17 that represents the category of the web page that was loaded. For instance the sequence $\{2,4,4,4,3\}$ means that the user loaded the "news" section, then loaded three times the "local" news and finally loaded the "technology" section.

The original work presented at (Guns et al., 2011), used an implementation based in Essence language (Frisch et al., 2008), but we have adopted a solution using MiniZinc (Nethercote et al., 2007) because it allows the specification of models using natural mathematical-like notation, speeding up our work.

6.2 Multisets Mining Resuts

As the original data from MSNCB.com contains repeated items per each transaction, so it can be used to test our multiset model. First of all the database needs to be flattened into the matrix representation shown in Section 4. That means that given the 17 different possible categories, a multiset $\{2,4,4,4,3\}$ needs to be transformed into $\{0,1,1,3,0,$
$0,0,0,0,0,0,0,0,0,0,0,0,0\}$. On the other hand, for our testing experiments we have used only the first 100 data entries.

Table 6 shows some of the patterns found. The most common behavior found within the explored data is to open the web front page, since 51% of users loads it. Opening the front page is a rather common

way to start the browsing, but with our extension we can know how many of them revisit this front page. Results show that 32% of users during its web exploration open a second time the front page and also that a 21% opens it three times. If we don't mind to reduce the support, we can find that 10% of explored users open twice the front page, twice again the technology page and also read the msn-sports.

Using the original approach, these patterns would have been found too, but without knowing how many times the users visit the pages. This proofs that there is still more information to mine than just the common pages browsed in the same session. Despite that we have worked with little subset of information, the conclusions of our results can be considered quite similar to previous works focused in the visualization of the sequences with this database (Cadez et al., 2000): after the front page, tech an local sections recive a lot of user visits.

6.3 Sequence Mining Results

Regarding sequence mining, the MSNBC data we have used only the first 100 data lines.

In Table 7 we present some of the patterns found. In this case the support is not as high as in the previous example, but the sequences are still interesting. The most common sequence followed by the 5% of the explored users consists in loading first msn-sports and then the sports ones, which are obvious related making a sensible pattern. The second most common sequence is also reasonable, since 4% of the users first open the general news web page and after that the local news.

We have to take under consideration that the found patterns are quite short (no more than two consecutive web sections) and that the support for these patterns is not very hight (no more than 5%). That can be caused by the original transformation we did in order to remove the repeated items, but still sustain the point that there was still information that could not be mined using the existing constraint-based approaches. In this case our transformation makes more difficult to compare our results with the ones at (Cadez et al., 2000), but still there are some patters in common like the $\{misc, local\}$.

6.4 Discussion

On one hand the experimentatrion done until now has shown that the running time for these models depends exclusively on the solver used. Minizinc has the capability of being a high level constraint programming language what allows the users to change its underlying solver easyly. Therefore, the implementation of the models "as they are" in this paper resulted in processing a set of data with a multipurpose solver for several hours while a fully optimized one returned the solutions with only less than a second of execution. Since the scope of this work does not involve surveying different solver implementations we relegate this task for future works. Another future work regarding process optimization may be the reification of the already known contraints in order to simplyfy the solver's work allowuing it to work even faster.

Table 7: Sequence patterns.

Support	Pattern
3%	frontpage, business
3%	local, health
3%	on-air, msn-news
3%	frontpage, local
3%	frontpage, news
3%	tech, local
3%	misc, local
3%	on-air, misc
4%	news, local
5%	msn-sports, sports

On the other hand, the results presented in this section show the feasibility of the constraint programming paradigm for multiset and sequence learning. Further experiments should be performed in other scenarios, and particularly with big data to find out the limits of the methodology. However, the important issue is that the learning process is not encapsulated in a complex and static procedure, but provided in a declarative way. This will allow the addition of new constraints without the need of modifying the base model, for instance, looking for closed or maximal patterns instead of frequent.

7 CONCLUSIONS

In this paper we have explored some of the state-of-the-art constraint-based mining and we have presented the modifications done to (Guns et al., 2011) that extend the original model so now it can support multiset and sequence mining.

Our model for multiset mining requires a change on the definition of the coverage constraint that takes into account the amount of items in the frequent pattern. The second, sequence mining, has the handicap of requiring more constraints in order to return valid outputs but the learning problem is provided in a declarative and simple way. We have shown examples of its application into toy problems to easily comprehend how the constraints works. Moreover, we

have applied the constraint programming models to the MSNBC database from the UCI repository, having successful results.

Our models where simple enough to work with academic examples, but further experimentation is required, focussing on escalation (both: length of transaction and database size). In this sense, our research is directed towards the use of reified constraints that optimize the constraint programming model.

Note that we have presented two independent models: one for mining item repetitions within itemsets and another one or mining the itemsets with order relations. The combination of both can lead us into a complete sequence patter mining. Once the combination is done, this model will be equiparable to classical algorithms used nowadays like prefixspan or clospan with the advantage of being declarative and easily extensible.

ACKNOWLEDGEMENTS

This work is supported by the research projects CTQ2008-06865-C02-02 and DPI2011-24929, and the grant FPU-AP2009-2831, all of them funded by the Spanish Government. The MSNBC dataset has been extracted from the UCI Machine Repository (Frank and Asuncion, 2010).

REFERENCES

(2012). Frequent itemset mining dataset repository. http://fimi.ua.ac.be/data/.

Agrawal, R. and Srikant, R. (1994). *Fast algorithms for mining association rules*, volume 1215, pages 487–499. Morgan Kaufmann.

Agrawal, R. and Srikant, R. (1995). Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering*, ICDE '95, pages 3–14, Washington, DC, USA. IEEE Computer Society.

Bonchi, F., Giannotti, F., Lucchese, C., Orlando, S., Perego, R., and Trasarti, R. (2009). A constraint-based querying system for exploratory pattern discovery. *Information Systems*, 34(1):3–27.

Bonchi, F. and Lucchese, C. (2007). Soft constraint-based pattern mining. *Data Knowl. Eng.*, 60:377–399.

Brand, M. (1998). Pattern discovery via entropy minimization. Technical report, MERL - A Mitsubishi Electric Research Laboratory.

Burke, R. (1999). The wasabi personal shopper: a case-based recommender system. In *Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*, AAAI '99/IAAI '99, pages 844–849, Menlo Park, CA, USA. American Association for Artificial Intelligence.

Cadez, I., Heckerman, D., Meek, C., Smyth, P., and White, S. (2000). Visualization of navigation patterns on a web site using model-based clustering. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '00, pages 280–284, New York, NY, USA. ACM.

David, J. and Nourine, L. (Submitted to Theoretical Computer Science). A generic algorithm for sequence mining. 15 pages. http://www-lipn.univ-paris13.fr/ david/old.version/articles/generation.pdf [Accessed: 9.2.2012].

De Raedt, L., Guns, T., and Nijssen, S. (2008). Constraint programming for itemset mining. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 204–212, New York, NY, USA. ACM.

De Raedt, L., Guns, T., and Nijssen, S. (2010). Constraint programming for data mining and machine learning. In *AAAI*, pages 1671–1675.

De Raedt, L. and Kramer, S. (2001). The levelwise version space algorithm and its application to molecular fragment finding. In *Proceedings of the 17th international joint conference on Artificial intelligence - Volume 2*, pages 853–859, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Frank, A. and Asuncion, A. (2010). UCI machine learning repository. http://archive.ics.uci.edu/ml.

Frisch, A., Harvey, W., Jefferson, C., Martinez-Hernandez, B., and Miguel, I. (2008). Essence: A constraint language for specifying combinatorial problems. *Constraints*, 13:268–306. 10.1007/s10601-008-9047-y.

Gal, Y., Reddy, S., Shieber, S. M., Rubin, A., and Grosz, B. J. (2012). Plan recognition in exploratory domains. *Artificial Intelligence*, 176(1):2270 – 2290.

Guns, T., Nijssen, S., and Raedt, L. D. (2011). Itemset mining: A constraint programming perspective. *Artificial Intelligence*, 175(12-13):1951 – 1983.

Han, J., Cheng, H., Xin, D., and Yao, X. (2007). Frequent pattern mining: current status and future directions. *Data Min Knowl Disc*, 15:55–86.

Nethercote, N., Stuckey, P. J., Becket, R., Brand, S., Duck, G. J., and Tack, G. (2007). Minizinc: towards a standard cp modelling language. In *Proceedings of the 13th international conference on Principles and practice of constraint programming*, CP'07, pages 529–543, Berlin, Heidelberg. Springer-Verlag.

Schmidt, C., Sridharan, N., and Goodson, J. (1978). The plan recognition problem: An intersection of psychology and artificial intelligence. *Artificial Intelligence*, 11(1-2):45 – 83. Applications to the Sciences and Medicine.

Soulet, A., Kléma, J., and Cremilleux, B. (2006). Efficient mining under flexible constraints through several datasets. In Džeroski, S. and Struyf, J., editors, *Proceedings of 5th International Workshop on Knowledge Discovery in Inductive Databases*, pages 131–142, Berlin, Germany. Humbolt Universität Berlin, 2006, s.

Srikant, R. and Agrawal, R. (1996). Mining sequential patterns: Generalizations and performance improvements. In Apers, P. M. G., Bouzeghoub, M., and Gardarin, G., editors, *Proc. 5th Int. Conf. Extending Database Technology, EDBT*, volume 1057, pages 3–17. Springer-Verlag.

Zhang, Z., Kwok, J. T., and Yeung, D.-Y. (2003). Parametric distance metric learning with label information. In *Proceedings of the 18th international joint conference on Artificial intelligence*, pages 1450–1452, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.