Mining rare sequential patterns with ASP

Ahmed Samet[1], Thomas Guyet[2], and Benjamin Negrevergne[3]

[1] Université Rennes 1/IRISA-UMR6074
[2] AGROCAMPUS-OUEST/IRISA-UMR6074
[3] LAMSADE, Université Paris-Dauphine

Abstract. This article presents an approach of meaningful rare sequential pattern mining based on the declarative programming paradigm of Answer Set Programming (ASP). The setting of rare sequential pattern mining is introduced. Our ASP approach provides an easy manner to encode expert constraints on expected patterns to cope with the huge amount of meaningless rare patterns. Encodings are presented and quantitatively compared to a procedural baseline. An application on care pathways analysis illustrates the interest of expert constraints encoding.

Keywords: rare patterns, sequential patterns, declarative pattern mining, patient care pathways

1 Introduction

Pattern mining aims at extracting meaningful structured knowledge hidden in large amounts of raw data. Building on the hypothesis that an interesting pattern occurs frequently in the data, most research on pattern mining have focused on *frequent* pattern mining. However, in many cases, *rare* patterns can also be meaningful. For example, this is true when physicians want to identify dangerous outcomes out of ordinary procedures from care pathway data. Fortunately for patients, such outcomes are rares but, unfortunately for data analysts, such events are difficult to extract using standard approaches.

Mining rare patterns has been studied in the context of itemsets [1]. But to the best of our knowledge, the problem of mining rare sequential patterns has not been addressed yet. The lack of work on this topic has been recently identified by Hu et *al.* [2] as an important matter for future research.

The main problem with rare patterns, known from experiments on rare itemsets, is their huge number. Condensed rare patterns, called "minimal rare patterns" [3], have been proposed to reduce the number of patterns to extract without loosing information. Yet, it is desirable to further reduce the number of patterns and to improve the pattern significance. The approach we propose in this paper, is to let the expert express extra constraints to specify the most interesting patterns. To achieve this goal, we need to develop a method to extract condensed rare sequential patterns which will be versatile enough to support easy addition of expert constraints. However, most approaches based on procedural programs would require specific and long developments to integrate extra

constraints. Declarative programming appears to be an interesting alternative to extract knowledge from data under complex expert constraints.

Using declarative programming, and more especially logic programming appears to be an interesting alternative to dedicated procedural algorithm several decades ago and received a new recent interests with the improvement of solver efficiency. The state-of-the-art can be organized according to two analysis axis: the data analysis tasks and the declarative programming paradigm. Data analysis task are classically separated in supervised *vs* unsupervised learning. More especially, inductive logic programming (ILP) [4] belongs to the field of supervised machine learning, while the more recent field of declarative pattern mining [5], which consists in mining patterns with declarative encodings, belongs to the field of unsupervised learning. For these two tasks, several declarative paradigms have been proposed: logic programming, Prolog or Answer Set Programming (ASP), Constraint Programming (CP) and Satisfiability Solving (SAT). For each approach, the objective is to propose a uniform representation technique for examples, background knowledge and hypotheses. ILP started with encodings in Prolog [4], and alternative systems have been implemented based on CP [6] or on ASP [7]. In declarative pattern mining, most of the approaches have been implemented using SAT [8], CP [9,10] and some with ASP [11]. In this work, we choose the paradigm of ASP to implement our rare sequential pattern mining to benefit from a first-order logic programming language. It makes the encoding of extra constraints easier and its solvers have proved their efficiency [11].

The contributions of this paper are threefold: *(i)* we formalize the general problem of rare sequential pattern mining and we model it in ASP, together with two important variations of this problem. Thanks to the flexibility of ASP, we were able to solve all three problems with the same ASP solver, avoiding the tedious work of designing and implementing an algorithm for each new problem. *(ii)* We provide important insights on modelling efficiency by comparing several alternative models. The experimental comparison shows that general ASP-based approaches can compete with ad-hoc specialized algorithms. And 3), we apply rare sequential mining to our target application of analyzing care pathway data. In particular, we demonstrate the benefit of additional constraints to extract meaningful results in this domain.

2 Rare sequential patterns: definitions and properties

This section introduces the basic concepts of rare sequential pattern mining.

Let $\mathcal{I} = \{i_1, i_2, \ldots, i_{|\mathcal{I}|}\}$ be a set of *items*. A *sequence* $\boldsymbol{s}$, denoted by $\langle s_j \rangle_{1 \leq j \leq n}$ is an ordered list of items $s_j \in \mathcal{I}$.

Given two sequences $\boldsymbol{s} = \langle s_j \rangle_{1 \leq j \leq n}$ and $\boldsymbol{s'} = \langle s'_j \rangle_{1 \leq j \leq m}$ with $n \leq m$, we say that $\boldsymbol{s}$ is a *subsequence* of $\boldsymbol{s'}$, denoted $\boldsymbol{s} \preceq \boldsymbol{s'}$, iff there exists n integers $i_1 < \ldots < i_n$ such that $s_k = s'_{i_k}$, $\forall k \in \{1, \ldots, n\}$.

Let $\mathcal{D} = \{\boldsymbol{s}^1, \boldsymbol{s}^2, \ldots, \boldsymbol{s}^N\}$, be a dataset of N sequences. The *support* of any sequence $\boldsymbol{s}$, denoted $supp(\boldsymbol{s})$, is the number of sequences $s^i \in \mathcal{D}$ that contain $\boldsymbol{s}$:

$$supp(\boldsymbol{s}) = \left|\left\{\boldsymbol{s}^i \in \mathcal{D} | \boldsymbol{s} \preceq \boldsymbol{s}^i\right\}\right| \quad (1)$$

	$\boldsymbol{s}^1$	$\boldsymbol{s}^2$	$\boldsymbol{s}^3$	$\boldsymbol{s}^4$	$\boldsymbol{s}^5$	$\boldsymbol{s}^6$	$\boldsymbol{s}^7$
Seq.	$\langle d, a, b, c\rangle$	$\langle a, c, b, c\rangle$	$\langle a, b, c\rangle$	$\langle a, b, c\rangle$	$\langle a, c\rangle$	$\langle b\rangle$	$\langle c\rangle$

Table 1. Example of artificial dataset of 7 sequences.

Let $\sigma \in \mathbb{N}^+$ be a frequency threshold given by the data analyst. A sequence $\boldsymbol{s}$ is *rare* iff $supp(\boldsymbol{s}) < \sigma$. In such case, the sequence $\boldsymbol{s}$ is called a rare sequential pattern or a *rare pattern* for short.

It is noteworthy that rare patterns are strongly related to the frequent patterns, *i.e.* the sequence $\boldsymbol{s}$ such that $supp(\boldsymbol{s}) \geq \sigma$. The set of rare patterns is the dual of frequent patterns. A pattern that is not rare is frequent and conversely.

Moreover, the rarity constraint is monotone, *i.e.* for two sequences $\boldsymbol{s}$ and $\boldsymbol{s}'$, if $\boldsymbol{s} \preceq \boldsymbol{s}'$ and $\boldsymbol{s}$ is rare, then $\boldsymbol{s}'$ is rare. This property comes from the anti-monotonicity of the support measure.

Example 1 (Sequences and rare patterns). Table 1 presents a simple dataset $\mathcal{D}$, defined over a set of items $\mathcal{I} = \{a, b, c, d\}$. The support of sequence $\boldsymbol{p}_1 = \langle a, b\rangle$ is 4 and the support of $\boldsymbol{p}_2 = \langle c, c\rangle$ is 1. For a support threshold $\sigma = 2$, $\boldsymbol{p}_2$ is a rare pattern but $\boldsymbol{p}_1$ is not. Due to the monotonicity of rarity, every extension of $\boldsymbol{p}_2$ is also rare (*e.g.* $\langle a, c, c\rangle$).

We now introduce two additional definitions: *zero-rare* patterns and *minimal rare patterns*. A *zero-rare* pattern is a rare pattern with a null support. This means that this pattern does not occur in dataset sequences. A *non-zero-rare* pattern is a rare pattern which is not zero-rare.

Finally, a sequence $\boldsymbol{s}$ is a minimal rare pattern (mRP) iff it is rare but all its proper subsequences are frequent.

$$mRP = \{\boldsymbol{s} | supp(\boldsymbol{s}) < \sigma \wedge \forall \boldsymbol{s}' \prec \boldsymbol{s}, supp(\boldsymbol{s}') \geq \sigma\} \tag{2}$$

Example 2 (minimal rare patterns). Figure 1 illustrates the lattice of non-zero rare patterns in the dataset $\mathcal{D}$ of Table 1 for $\sigma = 2$. $\langle b, a\rangle$ (not figured out in the lattice) is an example of zero-rare pattern insofar as it does not appear in $\mathcal{D}$. $\langle a, c, b\rangle$ is a non-zero-rare pattern. However, it is not an mRP since $\langle c, b\rangle$, one of its subpatterns (*i.e.* $\langle c, b\rangle \prec \langle a, c, b\rangle$), is rare.

3 Mining rare sequential patterns with ASP

In this section, we detail ASP encodings for mining rare and minimal rare sequential patterns. ASP is a declarative programming paradigm founded on the semantic of stable models [12]. From a general point of view, declarative programming gives a description of what is a problem instead of specifying how to solve it. A program describes the problem and a dedicated solver finds out its solutions.

ASP provides a high-level syntax that makes the program almost easy to understand compared to other declarative paradigms. An *ASP program* is a set

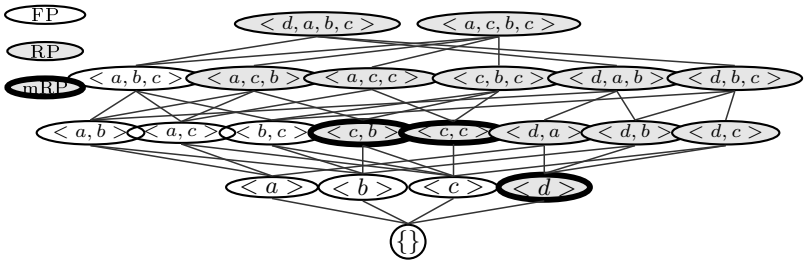

Fig. 1. Lattice of sequential patterns of dataset $\mathcal{D}$. Zero-rare patterns are omitted for sake of clarity. In white: frequent patterns, in grey: rare patterns, surrounded: minimal rare patterns for $\sigma = 2$.

```
seq(1,1,d). seq(1,2,a). seq(1,3,b). seq(1,4,c).
seq(2,1,a). seq(2,2,c). seq(2,3,b). seq(2,4,c).
seq(3,1,a). seq(3,2,b). seq(3,3,c).
seq(4,1,a). seq(4,2,b). seq(4,3,c).
seq(5,1,a). seq(5,2,c).
seq(6,1,b).
seq(7,1,c).
```

Listing 1.1. Facts specifying the sequence dataset of Table 1

of rules of the form $\mathtt{a_0}$ `:-` $\mathtt{a_1}, \ldots, \mathtt{a_m}, \mathtt{not}\ \mathtt{a_{m+1}}, \ldots, \mathtt{not}\ \mathtt{a_n}$, where each $\mathtt{a_i}$ is a propositional atom for $\mathtt{0} \leq \mathtt{i} \leq \mathtt{n}$ and `not` stands for *default negation*. Atoms can be encoded using a first-order logic syntax. If $\mathtt{n} = \mathtt{0}$, such rule is called a *fact*. If $\mathtt{a_0}$ is omitted, such rule represents an integrity constraint. The reader can refer to Janhunen et *al.* [12] for a more detailed introduction to ASP. Dedicated solvers, such as *clingo* [13], ensure the efficient solving of ASP encodings.

In the following, Section 3.1 introduces the ASP generation of candidate sequential patterns, then the rarity constraints is introduced in Section 3.2 and finally, Section 3.3 introduces additional constraints to extract efficiently only the minimal-rare patterns.

First of all, the dataset has to be encoded as facts. It is represented by atoms `seq(`s`,`t`,`i`)` stating that item i occurs at time t in sequence s. Listing 1.1 illustrates facts encoding of the sequence dataset of Table 1.

3.1 Enumerate candidate sequential patterns in ASP

Listing 1.2 specifies the candidate generation. It reuses some notations and principles introduced by Gebser et *al.* [11]. Line 1 lists all atoms present in the dataset. For each item $i \in \mathcal{I}$ a unique atoms `item(`i`)` is generated. The token `"_"` stands for an anonymous variable that does not recur anywhere. Line 2 defines the sequence lengths of each sequence from the dataset. The length of the sequence is an item position for which there is no items afterwards. Lines 4 to 7 enumerate all possible patterns. A pattern is a sequence of positions to fill up

with exactly one item[4]. It is encoded by atoms `pat(`x`,`i`)` where x is a position in the pattern and $i \in \mathcal{I}$. Predicate `patpos/1` takes one variable and defines the sequence position for pattern items. Rules with curly brackets in the head are choice rules such as lines 5 and 7. Rule at line 5 makes the pattern length choice which can not be larger than `maxlen`. Line 7 chooses exactly one atom `pat/2` for each position `X` and thus generates the candidate patterns combinatorics.

Lines 9 to 13 evaluate the pattern support. This encoding uses the fill-gaps strategy proposed in [11] (see Figure 2). The idea consists in mapping each sequence position to one (and exactly one) pattern position. In addition, some sequence positions are associated with an item that does not match the item in the mapped pattern position. This denotes filling positions and indicates which part of the pattern has already been "*read*". Lines 9 to 11 traverse each sequence $\boldsymbol{s} = \langle s_j \rangle_{1 \leq j \leq n}$ in $\mathcal{D}$ to yield atoms of the form `occ(`$\boldsymbol{s}$`,`l`,`p_l`)` where p_l is a position in the pattern and l is a position in the sequence $\boldsymbol{s}$. `occ(`$\boldsymbol{s}$`,`l`,`p_l`)` indicates that the p_l-th first items of the patterns have been read at position l of the sequence. Line 9 searches for a compatible position `P` with the first pattern item. Line 11 yields a new atom `occ(`$\boldsymbol{s}$`,`l`,`p_l`)` while the $(l-1)$-th item has been read at position $(p_l - 1)$ and the p_l-th sequence item matches the l-th pattern item. In any case, line 10 yields an atom associating position p_l with already read positions of the pattern. The `support(`$\boldsymbol{s}$`)` is yielded line 13 while sequence $\boldsymbol{t}$ holds the current pattern.

```
1 item(I) :- seq(_, _,I).
2 seqlen(S,L) :- seq(S,L,_), not seq(S,L+1,_).

4 patpos(1).
5 { patpos(X+1) } :- patpos(X), X<maxlen.
6 patlen(L) :- patpos(L), not patpos(L+1).
7 1 { pat(X,I) : item(I) } 1 :- patpos(X).

9 occ(S,1,P) :- pat(1,I), seq(S,P,I).
10 occ(S,L,P) :- occ(S, L,   P-1), seq(S,P,_).
11 occ(S,L,P) :- occ(S, L-1, P-1), seq(S,P,C), pat(L,C).

13 support(S) :- occ(S, L, LS), patlen(L), seqlen(S,LS).
```

Listing 1.2. Encoding of candidate patterns generation

For more details about fill-gaps strategy and its alternatives, we refer the readers to the article of Gebser et *al.* [11].

3.2 Efficient encoding of rarity constraint (ASP-RSM)

The modularity of ASP encoding enables to easily add constraints to the previous program. Where Gebser et *al.* added constraints to select only the frequent patterns, we select the answer sets that correspond to non-zero-rare patterns.

[4] We do not consider sequences of itemsets for sake of encoding simplification.

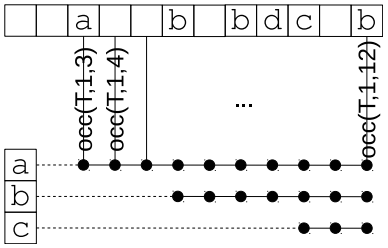

Fig. 2. Illustration of the fill-gaps embedding strategy (see Listing 1.2) for pattern $\langle abc \rangle$ in the sequence $\langle \dots a \dots b \dots bdc \dots b \dots \rangle$. Each black-circle illustrates one `occ/3` atoms in a model. ASP atoms of this predicate are detailed for three circles.

The following constraint states that the number of supported sequences must be strictly below the given threshold `th` (*i.e.* not above `th`). `th` is a program constant that can be specified while the solver is launched.

```
:- #count{ S : support(S) } >= th.
```

Despite the correctness of this constraint, we propose an alternative encoding that explicitly expresses the monotonicity property such that the solver may prune non-zero-rare patterns more efficiently. This encoding introduces a `rare/1` predicate (see listing 1.3). Instead of evaluating the support of pattern $\langle p_j \rangle_{1 \leq j \leq n}$, it evaluates independently the support of each of its prefixes. For any l, $1 \leq l \leq n$, `rare(`l`)` means that the subpattern $\langle p_j \rangle_{1 \leq j \leq l}$ is rare. Line 18 imposes to have the atom `rare(`n`)` where n is the pattern length. Line 17 gives a simpler manner to yield such atom: if a prefix-pattern of length l is rare, then all patterns of length l', $l' > l$, are rare. This rule prevents the solver from evaluating all rules of line 15 and it potentially prevents from evaluating `occ/3` atoms. Finally, line 19 prunes zero-rare patterns imposing to have at least one supported sequence.

```
15 rare(IL):- IL=1..L, patlen(L),
16            #count{ S : occ(S, IL, LS),seqlen(S,LS) } < th.
17 rare(L) :- rare(L-1), L<=PL, patlen(PL).
18 :- not rare(L), patlen(L).
19 :- not support(S) : seq(S,_,_).
```

Listing 1.3. Encoding of rare sequence mining

As the number of non-zero-rare patterns may be very high, the following section presents ASP-MRSM, an extension of ASP-RSM that extracts the condensed set of minimal-rare patterns, mRP.

3.3 ASP Minimal Rare Sequence Miner (ASP-MRSM)

The encoding in Listing 1.4 can be used to mine minimal rare sequences. It is based on the observation that it is sufficient to evaluate the support of all subpattern of size $n-1$ to determine whether a pattern $\boldsymbol{p}$ of size n is a minimal. Let $\boldsymbol{p} = \langle p_i \rangle_{1 \leq i \leq n}$ be a non-zero-rare pattern. $\boldsymbol{p}$ is a mRP iff $\forall u \in \{1, \dots, n\}$, the

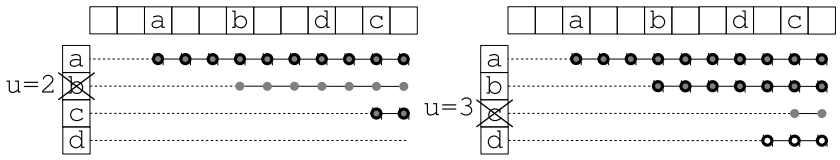

Fig. 3. mRP occurences for pattern $\boldsymbol{p} = \langle a, b, c, d\rangle$ and $\boldsymbol{u} = 3$, on the left, in case of a sequence that support $\boldsymbol{p}_u$ but not $\boldsymbol{p}$. Grey disks illustrate `occ/3` atoms and black circles illustrate `spocc/4`.

sub-pattern $\boldsymbol{p}_u = \langle p_i\rangle_{i\in\{1,\ldots,u-1,u+1,\ldots,n\}}$ is frequent (*i.e.* not rare). According to Equation 2, $\boldsymbol{p}$ is a mRP iff $\forall \boldsymbol{p}' \prec \boldsymbol{p}$, $\boldsymbol{p}'$ is frequent. It is then clear that if $\boldsymbol{p}$ is a mRP, then $\forall u$, $\boldsymbol{p}_u$ is frequent. Conversely, for all sequence $\boldsymbol{s}$ such that $\boldsymbol{s} \prec \boldsymbol{p}$, $\exists u$ such that $\boldsymbol{s} \prec \boldsymbol{p}_u$. Then, as $\boldsymbol{p}_u$ is frequent and by anti-monotonicity of the frequency constraint, $\boldsymbol{s}$ is frequent too.

Furthermore, the encoding takes advantage of two properties of sequential patterns: (1) a sequence that supports the pattern $\boldsymbol{p}$ supports each pattern $\boldsymbol{p}_u$, (2) if a sequence $\boldsymbol{s}$ is supported by $\boldsymbol{p}_u$ but not by $\boldsymbol{p}$, then for all $l \in [1, u-1]$, `occ(`$\boldsymbol{s}$`,`l`,`p_l`)` is part of the ASP model computed by Listing 1.2, where $(p_l)_{l\in[1,(n-1)]}$ is an occurrence of $\boldsymbol{p}_u$. In fact, as there is no difference between $\boldsymbol{p}_u$ and $\boldsymbol{p}$ before position u, there is no differences between their occurrences. These properties are helpful to decide whether $\boldsymbol{p}_u$ is rare. In Listing 1.4, line 20 enumerates all subpatterns, identified by `U`. Similarly to `occ/3` predicates, atoms `spocc(`$\boldsymbol{s}$`,`u`,`l`,`p_l`)` describe the occurrence of $\boldsymbol{p}_u$ in sequence $\boldsymbol{s}$ (Figure 3).

Lines 22 to 26 compute occurrences of pattern $\boldsymbol{p}_u$ according to the fill-gaps principle (see section 3.1) with two differences: (1) `spocc/4` are yielded only if sequence $\boldsymbol{t}$ is not supported, otherwise the sequence is obviously supported knowing that $\boldsymbol{p}_u \prec \boldsymbol{p}$; and (2) `spocc/4` starts from pattern position u using `occ(`$\boldsymbol{t}$`,`$u-1$`,_)` available atoms to avoid redundant computation according to the second property above. Lines 28 to 33 determine whether the subpattern u is rare. This encoding differs from section 3.2 mainly at lines 30-32. The total number of sequences supported by $\boldsymbol{p}_u$ is the number of sequences supported by $\boldsymbol{p}$ (line 32) plus the number of sequences supported by $\boldsymbol{p}_u$ but not by $\boldsymbol{p}$ (line 31), *i.e.* sequences for which `spocc/4` atoms reached the last pattern item. Finally, line 33 eliminates answer sets for which a subpattern is rare.

It is worth noticing that in ASP two answer sets can not share information during the solving process. Gebser et *al.* [11] proposed to use a solver extension, called *asprin*, to encode the mining of closed patterns, *i.e.* patterns which have any larger patterns with the same support. A similar approach may be used for mRP. Our approach prefers a pure declarative but more efficient encoding.

4 Experiments and results

In this section, we evaluate our approach on synthetic and real datasets. First, we use synthetic datasets to compare performances of our ASP encodings with

```
20 suppat(U) :- U=1..L, patlen(L), L>1.

22 spocc(S,1,2,P) :- seq(S,P,I), pat(2,I), not support(S).
23 spocc(S,U,U,P) :- suppat(U), occ(S,U-1,P), not support(S).
24 spocc(S,U,L  ,P+1) :- spocc(S,U,L,P), seq(S,P+1,_).
25 spocc(S,U,L+1,P+1) :- spocc(S,U,L,P), seq(S,P+1,I),
26                       pat(L+1,I).

28 sprare(U,U-1):- sprare(U-1), suppat(U).
29 sprare(U,L)  :- sprare(U, L-1), L<=LP, patlen(LP).
30 sprare(U, IL):- suppat(U), IL=U+1..L, patlen(L),
31                 #count{ S: spocc(S,U,IL,LS), seqlen(S,LS);
32                         S: support(S) } < th.
33 :- sprare(U,L), patlen(L).
```

Listing 1.4. Encoding part for minimal rare patterns

procedural algorithms. Then, we demonstrate the flexibility of our declarative pattern mining approach on a case study.

In all experiments, we used *clingo*, version 5.0. All programs were run on a computing server with 16Gb RAM. Multi-threading capabilities of *clingo* have been disabled because existing procedural algorithms are not parallel and because multi-threading introduces high variance in run times measurements.

4.1 Runtime and memory efficiency on synthetic datasets

In this first set of experiments, we use synthetic datasets to compare the efficiency of our ASP encodings with procedural algorithms. Four approaches have been compared: ASP-RSM and Apriori-Rare extract non-zero-rare patterns; and ASP-MRSM and MRG-EXP extract minimal rare patterns. Since Apriori-Rare [3] and MRG-Exp [14], were originally developed for rare itemset mining, they were adapted to mine rare sequential patterns[4].

We use several synthetic datasets to measure the impact of various parameters such as *vocabulary size* and *means sequence length* on the runtime and on the memory usage. Our sequence generator[4] produces datasets following a 3-step *retro-engineering* procedure inspired from the IBM Quest Synthetic Data Generator: 1) a set of random patterns is generated, 2) for each pattern, a list of occurrences is generated, and 3) synthetic sequences are built so that each sequence contains a pattern if the sequence is in the pattern occurrence list. The sequences are then completed with random items so that the mean sequence length of the resulting dataset reaches the input parameter λ.

[4] The source codes of the ASP programs, of procedural algorithms and of the dataset generator are available at `http://www.ilp-paper-1.co.nf/`

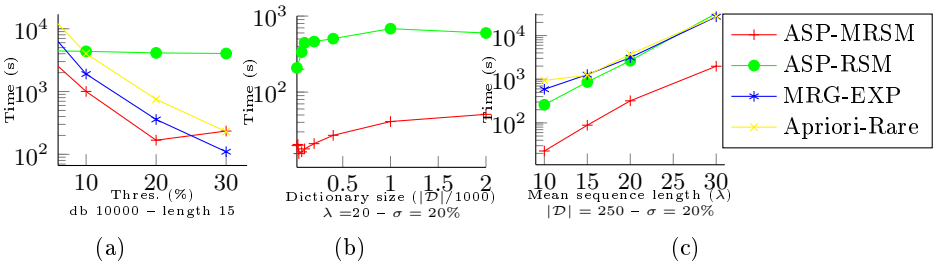

Fig. 4. (a) is computing times *w.r.t.* threshold for synthetic databases of size 10^5 (with a mean sequence length of 15). (b) and (c) are mean computing times *w.r.t.* sequence length and vocabulary size (with $|\mathcal{D}| = 250$ and $\sigma = 20\%$).

Threshold (σ)	#patterns		Memory usage (bytes)			
	#Rare	#mRP	ASP-RSM	ASP-MRSM	Apriori-Rare	MRG-EXP
Dataset – 1000 sequences						
5%	1 011 117	260	1.0×10^8	0.7×10^8	2.5×10^6	5.1×10^6
10%	1 011 993	527	1.0×10^8	0.7×10^8	5.3×10^6	7.5×10^6
20%	1 012 360	2 280	1.1×10^8	0.7×10^8	9.6×10^6	1.2×10^7
30%	1 012 428	7 123	1.3×10^8	0.9×10^8	9.8×10^6	5.3×10^7
Dataset – 5000 sequences						
5%	105 907	477	1.0×10^8	3.5×10^8	5.2×10^5	5.8×10^5
10%	109 861	957	1.0×10^8	3.5×10^8	5.2×10^6	7.3×10^5
20%	110 879	4 754	1.0×10^8	3.7×10^8	5.6×10^6	3.7×10^6
30%	111 022	16 530	1.2×10^8	4.8×10^8	7.8×10^6	3.8×10^7
Dataset – 10000 sequences						
5%	108 266	456	2.0×10^8	6.5×10^8	1.3×10^6	1.3×10^6
10%	110 334	734	2.0×10^8	6.5×10^8	3.4×10^6	3.1×10^6
20%	110 955	4 086	2.0×10^8	6.5×10^8	5.6×10^6	3.6×10^6
30%	111 036	10 177	2.0×10^8	8.0×10^8	7.2×10^6	7.3×10^6

Table 2. Memory consumption and number of patterns of ASP-RSM, ASP-MRSM, Apriori-Rare and MRG-EXP *w.r.t* dataset size and rarity threshold (with $\lambda = 10$).

Figure 4 presents runtimes for variable dataset size (a), variable sequence length (*b*), and variable vocabulary size (*c*). Table 2 presents the number of extracted patterns and the memory footprint.

First, we can see in Figure 4-(a) that ASP-MRSM is an order of magnitude faster than alternatives approaches. This is an important result since previous work on declarative pattern mining have shown that declarative approaches are often slower than the procedural counterparts [11,15]. This experiment demonstrates that the overhead induced by the declarative encoding is balanced by the more efficient solving strategy achieved by the ASP solver.

Second, we focus on ASP-RSM and Apriori-Rare (non-zero-rare-patterns). In general, decreasing the threshold induces a larger search space and thus higher runtimes. However, we can see that ASP-RSM is far less sensitive to variations of the threshold than Apriori-Rare is. As a consequence, ASP-RSM is faster for very low thresholds. Apriori-Rare is greatly impacted by the increasing search space, even though the number of rare patterns remains almost constant as it

can be seen in Table 2. In contrast the runtime of ASP-RSM remains fairly low, even when the threshold is low. This demonstrates again that the ASP solver is able to explore the search space more efficiently.

Figure 4-(b) shows that both declarative and procedural approaches are penalised by larger sequence lengths. The algorithm behaviour on this plot is consistent with previous plots and ASP-MRSM remains the most efficient approach.

Figure 4-(c) shows the mean computing time with an increasing vocabulary size $|\mathcal{I}|$. Understanding the impact of the size of the vocabulary is not trivial, since a smaller vocabulary means fewer items to process but also increases the dataset density. In Table 2, we can see that the fewer are items, the fewer are rare patterns (with a fixed size, $|\mathcal{D}|$ and mean length, λ). Remark that the slope decreases with the vocabulary size: the number of patterns increases quickly for small vocabularies, but increases slowly for larger vocabularies. This behaviour is explained by the combinatorics of items: with a lot of rare items, the mean length of mRP is smaller because it is more probable to have patterns of length 1 that are non-zero rare (with our simulated dataset). In fact, the shorter is an mRP, the more rare patterns it represents. As a consequence, the computation time increases less faster.

4.2 Application to rare care pathways analytics

We now apply our approach to care pathway analytics to illustrate that the number of rare patterns can be reduced using additional expert constraints. This is not a new result but we show that extracting rare patterns becomes practically interesting with constraints. The ASP approach was here crucial to incorporate constraints easily. Care pathway analytics is a research field that consists in exploring patient care pathways to understand care uses, more especially drugs consumption in this case, and their impacts on patients health.

This case study analyzes care pathways of patients exposed to anti-epileptic drugs who had epileptic seizures. Rare pattern mining aims at identifying (i) patients to exclude from the study because of their unexpected rare care pathways, for instance due to specific heavy treatments (cancer, hepatitis C, AIDS, etc.); (ii) rare adverse drugs reactions. The dataset is made of 500 care pathways with drugs deliveries during one year. This represents a total amount of $101,793$ events, with $|\mathcal{I}| = 3,671$. Drug deliveries are encoded with `seq(t,p,e)` atoms where e is the active component of the delivered drug.

The rough mining of minimal rare sequential patterns with $\sigma = 10\%$ yields $7,758$ mRPs of length at most 3 ($maxlen = 3$). This pattern number makes these results difficult to analyze by clinicians.

Two types of constraints have been added in order to illustrate the pattern reduction. The first one is a relaxation of rare patterns. Too rare patterns are poorly significant but so numerous. We thus introduced a second parameters f_{min} such that the support of extracted patterns must be between f_{min} and σ. The second one concerns the maximum duration between two events of a pattern occurrence. This constraint, known as the *maxgap* constraint, avoids to

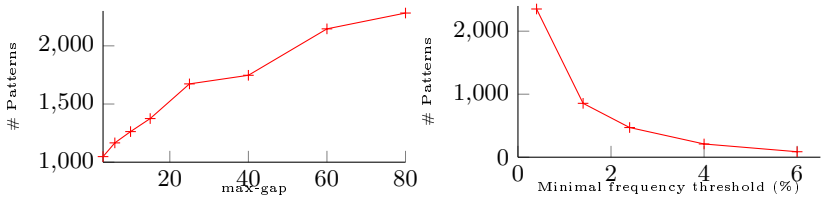

Fig. 5. Number of rare patterns extracted from care pathways *w.r.t* expert constraints.

enumerate an occurrence while two of its events are too far in time. It assumes that the first event may causally be related to the second one.

Figure 5 illustrates the efficiency of this constraint to reduce the number of rare patterns with $\sigma = 10\%$ and $maxlen = 3$. The plot on the left illustrates the number of patterns *w.r.t.* the $maxgap$ constraints value (from 3 to 25 events). Considering that having larger max-gap reduces the number of non-zero patterns, this number increases with $maxgap$. The plot on the right illustrates the number of patterns *w.r.t.* the minimal frequency constraint, f_{min} (from 0.4% to 6%). The closer f_{min} is to σ, the less the number of patterns. This number decreases exponentially with f_{min}. The number of patterns extracted with $f_{min} = 6\%$ fall down to 87. This number of patterns is sufficiently low to be presented to clinicians. Moreover, the computing time is reduced by using expert constraints. This final experiment is solved in $425s$ which is significantly faster than in our experiments on synthetic datasets.

The two constraints can be combined. We choose to select patterns with $f_{min} = 5\%$ and $maxgap = 3$. In such case, the number of patterns fall down to 133. One interesting pattern describes the deliveries of *lamotrigine*, an anti-epileptic drug, co-occurring with two antibiotics *ceftriaxone* and *amoxicillin*. This unexpected potential adverse drug reaction may lead to new epidemiological questions.

5 Conclusion

This article extends existing work, notably Gebser et *al.* [11] work, on declarative sequential pattern mining with Answer Set Programming by proposing efficient encoding of rare patterns. It shows the versatility of declarative approach to design easily a new data mining task. We shown, on the one hand, that the ASP encodings appears to be an efficient solution compared to procedural approaches, but that its memory consumption limits the analysis to mid-size databases. This problem is known in pattern mining with ASP [11] and the solution will be to use dedicated propagators [15] to improve computing efficiency of the approach. On the other hand, we illustrate that this approach answers an applicative question by extracting sequential patterns satisfying additional expert constraints. We shown that constraints prune efficiently useless patterns with benefit on computation time.

Acknowledgements This work is a part of the PEPS project funded by the French national agency for medicines and health products safety (ANSM), and of the SePaDec project funded by Region Bretagne.

References

1. Koh, Y.S., Ravana, S.D.: Unsupervised rare pattern mining: A survey. Transactions on Knowledge Discovery from Data **10**(4) (2016) 1–29
2. Hu, Z., Wang, H., Zhu, J., Li, M., Qiao, Y., Deng, C.: Discovery of rare sequential topic patterns in document stream. In: Proceedings of the SIAM International Conference on Data Mining. (2014) 533–541
3. Szathmary, L.: Finding minimal rare itemsets with an extended version of the Apriori algorithm. In: Proceedings of the International Conference on Applied Informatics. Volume 1. (2014) 85–92
4. Muggleton, S., de Raedt, L.: Inductive logic programming: Theory and methods. The Journal of Logic Programming **19** (1994) 629 – 679
5. De Raedt, L., Guns, T., Nijssen, S.: Constraint programming for itemset mining. In: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM (2008) 204–212
6. Sebag, M., Rouveirol, C.: Constraint inductive logic programming. In: Advances in Inductive Logic Programming. IOS Press (1996) 277–294
7. Corapi, D., Russo, A., Lupu, E.: Inductive logic programming in answer set programming. In: International Conference on Inductive Logic Programming, Springer (2011) 91–97
8. Jabbour, S., Sais, L., Salhi, Y.: Decomposition based SAT encodings for itemset mining problems. In: Proceeding of the Pacific-Asia Conference Advances on Knowledge Discovery and Data Mining, Part II. (2015) 662–674
9. De Raedt, L., Guns, T., Nijssen, S.: Constraint programming for data mining and machine learning. In: Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10). (2010) 1671–1675
10. Guns, T., Dries, A., Nijssen, S., Tack, G., Raedt, L.D.: Miningzinc: A declarative framework for constraint-based mining. Artif. Intell. **244** (2017) 6–29
11. Gebser, M., Guyet, T., Quiniou, R., Romero, J., Schaub, T.: Knowledge-based sequence mining with ASP. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI). (2016) 1497–1504
12. Janhunen, T., Niemelä, I.: The answer set programming paradigm. AI Magazine **37** (2016) 13–24
13. Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., Schneider, M.: Potassco: The Potsdam answer set solving collection. AI Communications **24**(2) (2011) 107–124
14. Szathmary, L., Valtchev, P., Napoli, A.: Generating rare association rules using the minimal rare itemsets family. International Journal on Software and Informatics **4**(3) (2010) 219–238
15. Negrevergne, B., Guns, T.: Constraint-based sequence mining using constraint programming. In: Proceedings of the International Conference on Integration of AI and OR Techniques in Constraint Programming. (2015) 288–305