A Declarative Approach to Contrast Pattern Mining

Francesca Alessandra Lisi[1][0000−0001−5414−5844] and Gioacchino Sterlicchio[2][0000−0002−2936−0777]

[1] Dipartimento di Informatica & CILA
University of Bari "Aldo Moro", Italy
FrancescaAlessandra.Lisi@uniba.it
[2] Dept. Mechanics, Mathematics & Management,
Polytechnic University of Bari, Italy
g.sterlicchio@phd.poliba.it

Abstract. This paper proposes a declarative approach to the problem of contrast pattern mining. The approach is based on encodings of the data and the problem with Answer Set Programming (ASP), and evaluated in a novel AI application in the field of Digital Forensics.

Keywords: Contrast Pattern Mining · Answer Set Programming · Digital Forensics.

1 Introduction

Pattern mining [12] is a class of data mining tasks that consist of extracting interesting structured patterns from a dataset. These tasks encompass itemset mining, sequence mining and graph mining. The interestingness measure of a pattern is, in most of the algorithms, the number of its occurrences in the dataset. Given a threshold k, interesting patterns are those that occur at least in k data instances. In this case, the task is known as *frequent pattern mining* for which many algorithms have been proposed. An interesting extension of the frequent pattern mining task is the one that aims at the discovery of so-called *contrast patterns*. Whereas frequent patterns are statistically significant regularities in a set of transactions, contrast patterns denote statistically significant differences between two or more disjoint sets of transactions [6].

Recently there has been an increasing interest in declarative approaches to pattern mining, thus giving rise to a novel stream of research known under the name of *Declarative Pattern Mining* (DPM). So far, DPM addressed tasks such as frequent itemset mining [13,10], and sequence mining [17,7]. Different declarative frameworks have been explored: SAT [13], Constraint Programming [5,10], and Answer Set Programming (ASP) [7,11]. In this paper we propose a declarative approach for contrast pattern mining which leverages the expressive and inferential power of ASP. To the best of our knowledge, this interesting class of pattern mining problems has not been addressed yet in DPM.

Declarative approaches are generally desirable in application domains where the requirements of transparency, verifiability and explainability of the AI techniques employed are of paramount importance. One of these cases is the field of *Digital Forensics* (DF), a branch of criminalistics that deals with the identification, acquisition, preservation, analysis and presentation of the information content of computer systems, or in general of digital devices, by means of specialized software, and according to specific regulations. A declarative approach to DF was first explored by Costantini *et al.* [2,3], and subsequently adopted by the COST Action "Digital forensics: evidence analysis via intelligent systems and practices" (DigForASP)[3]. The aim of DigForASP is to promote formal and verifiable AI methods and techniques in the analysis of evidence [4]. In this paper, we report the preliminary results obtained by applying the proposed ASP-encoded contrast pattern mining algorithm to a dataset of phone records made available within DigForASP.

The paper is organized as follows. In Section 2 we provide the necessary preliminaries on contrast pattern mining and ASP. In Section 3 we introduce the proposed ASP encoding for contrast pattern mining. In Section 4 we describe the application of this encoding to the analysis of phone records, and report the results of some experiments. In Section 5 we conclude with final remarks.

2 Preliminaries

2.1 Contrast pattern mining in brief

We assume the set $I = \{1, ..., m\}$ of m items, and the set $T = \{1, ..., n\}$ of n transactions. Intuitively, a transaction $t \in T$ is a subset of items from I, which is typically associated with a transaction identifier (TID). A transactional database $\mathcal{D} \in \{0,1\}^{n \times m}$ can be seen as a binary matrix, in which each row $\mathcal{D}_t$ represent the transaction t consisting of the items $\{i \in I | \mathcal{D}_{t,i} = 1\}$, where $\mathcal{D}_{t,i}$ denote the value on the i-th column and t-th row of $\mathcal{D}$. The subsets of I are called *itemsets* or patterns. In pattern mining we are interested in finding patterns that satisfy constraints relative to a set of transactions. In particular, given the pattern $P \subseteq I$, and a set of transactions T, the subset of T covered by P is $cover(P, T) = \{t \in T | \forall i \in P : \mathcal{D}_{t,i} = 1\}$. Then the *absolute support* of P in T is defined as:

$$supp(P, T) = |cover(P, T)| \tag{1}$$

and quantifies the number of transactions in T containing the pattern P.

Frequent pattern mining algorithms are used to discover statistically significant regularities in a set of transactions whereas the contrast pattern mining task is about detecting statistically significant differences (*contrast*) between two or more disjoint sets of transactions [6]. To this aim, we assume also a finite set L of class labels which are used by the function $\mathcal{L}(t) \in L$ to label each transaction t. In our setting, the label $\alpha \in L$ partitions T in two samples:

[3] https://digforasp.uca.es/

1. $T(\alpha) = \{t \in T | \mathcal{L}(t) = \alpha\}$, *i.e.*, the transactions labeled with α;
2. its complement $T'(\alpha) = T \setminus T(\alpha)$.

The contrast pattern P with respect to α is quantified by the so-called *absolute support difference*, which is defined as:

$$diff(P, \alpha) = supp(P, T(\alpha)) - supp(P, T'(\alpha)) \tag{2}$$

The problem of contrast pattern mining concerns the enumeration of all frequent patterns with absolute support difference that exceeds the user-defined minimum support threshold *minDiff*. More specifically, given:

- the transaction database $\mathcal{D}$ over the set of transactions T;
- the maximum pattern length threshold *maxLength*;
- the minimum absolute support threshold $minSupp \geq 0$;
- the minimum absolute support difference threshold $minDiff \geq 0$;
- the label $\alpha \in L$.

the problem of contrast pattern mining is to find all patterns $(P, diff(P, \alpha))$ such that:

1. $|P| \leq maxLength$;
2. $supp(P, T(\alpha)) \geq minSupp$;
3. $diff(P, \alpha) \geq minDiff$.

To understand the meaning of contrast patterns, it is important to comment further the formula (2). Given a class α, a pattern P is a contrast pattern for that class if its support differs from the support of the same pattern for the complementary class. If the difference of the support is equal to 0, it means that P is present in the same way in the two classes. Therefore this pattern does not allow to find the differences between the classes. Conversely, the more the difference in support moves away from 0, the more P is to be understood as a pattern that allows to distinguish the two classes under comparison. Therefore, P is a representative pattern for the class α but not for the complementary class.

2.2 Answer Set Programming in a nutshell

In the following we give a brief overview of the syntax and semantics of disjunctive logic programs in ASP. The reader can refer to, *e.g.*, [1] for a more extensive introduction to ASP.

Let U be a fixed countable set of (domain) elements, also called *constants*, upon which a total order $\prec$ is defined. An *atom* α is an expression $p(t_1, \ldots, t_n)$, where p is a predicate of arity $n \geq 0$ and each t_i is either a variable or an element from U (*i.e.*, the resulting language is function-free). An atom is *ground* if it is free of variables. We denote the set of all ground atoms over U by B_U. A *(disjunctive) rule* r is of the form

$$a_1 \vee \ldots \vee a_n \leftarrow b_1, \ldots, b_k, not\ b_{k+1}, \ldots, not\ b_m$$

with $n \geq 0$, $m \geq k \geq 0$, $n + m > 0$, where $a_1, \ldots, a_n, b_1, \ldots, b_m$ are atoms, or a count expression of the form $\#count\{l : l_1, \ldots, l_i\} \bowtie u$, where l is an atom and l_j is a literal (*i.e.*, an atom which can be negated or not), $1 \geq j \geq i$, $\bowtie \in \{\leq, <, =, >, \geq\}$, and $u \in \mathbb{N}$. Moreover, "not" denotes *default negation*. The *head* of r is the set $head(r) = \{a_1, \ldots, a_n\}$ and the *body* of r is $body(r) = \{b_1, \ldots, b_k, notb_{k+1}, \ldots, notb_m\}$. Furthermore, we distinguish between $body^+(r) = \{b_1, \ldots, b_k\}$ and $body^-(r) = \{b_{k+1}, \ldots, b_m\}$. A rule r is *normal* if $n \leq 1$ and a *constraint* if $n = 0$. A rule r is *safe* if each variable in r occurs in $body^+(r)$. A rule r is *ground* if no variable occurs in r. A *fact* is a ground rule with $body(r) = \emptyset$ and $|head(r)| = 1$. An *(input) database* is a set of facts. A *program* is a finite set of rules. For a program Π and an input database D, we often write $\Pi(D)$ instead of $D \cup \Pi$. If each rule in a program is normal (resp. ground), we call the program normal (resp. ground).

Given a program Π, let U_Π be the set of all constants appearing in Π. $Gr(\Pi)$ is the set of rules $r\sigma$ obtained by applying, to each rule $r \in \Pi$, all possible substitutions σ from the variables in r to elements of U_Π. For count-expressions, $\{l : l_1, \ldots, l_n\}$ denotes the set of all ground instantiations of l, governed through $l_1, \ldots, l_n$. An interpretation $I \subseteq B_U$ satisfies a ground rule r iff $head(r) \cap I \neq \emptyset$ whenever $body^+(r) \subseteq I$, $body^-(r) \cap I = \emptyset$, and for each contained count-expression, $N \bowtie u$ holds, where $N = |\{l | l_1, \ldots, l_n\}|$, $u \in \mathbb{N}$ and $\bowtie \in \{\leq, <, =, >, \geq\}$. A ground program Π is satisfied by I, if I satisfies each $r \in \Pi$. A non-ground rule r (resp., a program Π) is satisfied by an interpretation I iff I satisfies all groundings of r (resp., $Gr(\Pi)$). A subset-minimal set $I \subseteq B_U$ satisfying the *Gelfond-Lifschitz reduct* $\Pi^I = \{head(r) \leftarrow body^+(r) | I \cap body^-(r) = \emptyset, r \in Gr(\Pi)\}$ is called an *answer set* of Π. We denote the set of answer sets for a program Π by $AS(\Pi)$.

The tools used in this work are part of the Potassco[4] collection [9]. The main tool of the collection is the *clingo* ASP solver [8].

3 Mining contrast patterns with ASP

Within the declarative framework of ASP, the transaction database $\mathcal{D}$ is represented by means of facts of the following two kinds: *class(t, c)*, and *db(t, f(v))*. Here, t is the TID while c represents the class, f represents a feature and v its value. In particular, we introduce the fact *db(t,f(v))* if and only if $\mathcal{D}_{t,i} = 1$. So, there is a *db*-fact for each feature.

In DPM, patterns are represented as answer sets. More precisely, a single pattern is associated with each answer set and in our approach represented by means of the *in_pattern/1* and *absolute_diff/1* predicates. The latter expresses the difference in support of the pattern between the class under consideration and the complementary class. Each pattern conveys information that allows to characterize the considered class.

[4] https://potassco.org/

```
1  #const minSupp = 2.
2  #const maxLength = 3.
3  #const minDiff = 1.
4  #const class = positive.
5
6  % link facts to objects used in the encoding
7  item(I) :- db(_,I).
8  transaction(T) :- db(T,_).
9
10 % problem encoding (frequent itemset mining)
11 {in_pattern(I)} :- item(I).
12 in_support(T) :- {conflict_at(T,I) : item(I)} 0,
     transaction(T), class(T, class).
13 out_support(T) :- {conflict_out(T,I) : item(I)} 0,
     transaction(T), not class(T, class).
14 conflict_at(T,I) :- not db(T,I), in_pattern(I), transaction
     (T), class(T, class).
15 conflict_out(T,I) :- not db(T,I), in_pattern(I),
     transaction(T), not class(T, class).
16
17 % definition of absolute support difference (Dong et al.)
18 absolute_diff(D) :- N = #count{ T : in_support(T)}, M = #
     count{T : out_support(T)}, D = |N-M|.
19
20 % length constraint
21 :- maxLength+1 {in_pattern(I)}.
22 :- {in_pattern(I)} 0.
23
24 % frequency constraint
25 :- {in_support(T)} minSupp-2.
26
27 % absolute growth-rate constraint
28 :- absolute_diff(D), D < minDiff.
29
30 % print directives for an answer-set
31 #show in_pattern/1.
32 #show absolute_diff/1.
```

Listing 1.1: Full ASP encoding for contrast pattern mining.

The ASP enconding for the contrast pattern mining problem introduced in Section 2.1 is reported in Listing 1.1. The values for *minSupp*, *minDiff* and *maxLength* are encoded as symbolic constants. In Lines 1-4, the chosen constants are for demonstration purposes only. The predicate *in_pattern/1* (Line 11) is true for an item i if and only if i is included in a pattern P and encoding the most important part of a solution $(P, diff(P, \alpha))$. The predicate *in_support/1* (Line 12) is true for a transaction t if and only if $t \in T$. The intuition is that each t has to support each $i \in I$ in the sense that t must include i. Additionally, we use

the auxiliary predicates *item/1* (Line 7, true for each item in $\mathcal{D}$), *transaction/1* (Line 8, true for each transaction in $\mathcal{D}$) and *conflict_at/2* (Line 14) which is true for *(t, i)* if and only if t does not support i, that is, we have the conflict $\mathcal{D}_{t,i} = 0$ and $i \in I$, thus violating the premises. In particular, the predicates *in_support/1* and *conflict_at/2* encode the construction of patterns for the class α. Conversely, the predicates *out_support/1* (Line 13) and *conflict_out/2* (Line 15) are used to generate patterns for the complementary class. Finally, the definition for the absolute support difference is encoded at Line 18.

After the pattern generation step, the encoding applies three constraints corresponding to the thresholds *maxLength*, *minSupp*, and *minDiff*. The first constraint is expressed by Lines 21-22 and rules out patterns having 0 items or more than *maxLength* items. The second constraint is expressed at Line 25. In fact, patterns supported by at most *minSupp-2* instances are not allowed as an answer. The third constraint, encoded at Line 28, discards patterns with absolute support difference lower than *minDiff* from the answer set. The two *#show* commands on Lines 31-32 allow, for each answer set, the display of the atoms that compose a solution $(P, diff(P, \alpha))$ to problem in hand.

The encoding and further material can be found online.[5]

4 An Application in Digital Forensics

Digital Forensics (DF) is a branch of criminalistics that deals with the identification, acquisition, preservation, analysis and presentation of the information content of computer systems, or in general of digital devices, by means of specialized software, and according to specific regulations. In particular, the phase of *Evidence Analysis* involves examining and aggregating evidence about possible crimes and crime perpetrators collected from various electronic devices in order to reconstruct events, event sequences and scenarios related to a crime. Results from this phase are then made available to law enforcement, investigators, intelligence agencies, public prosecutors, lawyers and judges.

During the investigation of a crime, it is common to analyze the communications of a particular suspect. Since nowadays mobile phones are objects owned by anyone, it can be useful for investigators to analyze the calls or messages exchanged. The telephone records are a set of data relating to the external communications of the devices. In other words, they contain all the traces of communications (calls, SMS, and all the data traffic) concerning a specific user over a certain period of time. Note that phone records do not trace sensitive data such as the audio of calls sent or received. In fact, phone records only provide a trace of the communication that has taken place but not its content.

The phone records can be requested by the Judicial Authority if deemed useful in order to carry out investigations involving the individual owner of the phone. Correctly analyzing the telephone records is essential to obtain useful hints. Depending on the analysis, different kinds of information can be extracted.

[5] https://github.com/mpia3/Contrast-Pattern-Mining

The records are typically analyzed for comparing the geographical positions with respect to the declarations, and for reconstructing the network of contacts of a single user in order to trace which conversations (s)he has had with whom, where and when. In this Section we report the preliminary results obtained by applying our ASP encoding for contrast pattern mining to a dataset of phone records.

4.1 The DigForASP dataset

For our experiments we have considered a dataset that consists of the telephone records of four users from a real-world investigative case. The dataset has been made available by Prof. David Billard (University of Applied Sciences in Geneva) under NDA to DigForASP members for academic experimentation.

Each file in the dataset has the following schema:

- *Type*: what kind of operation the user has performed (*e.g.*, incoming/outgoing call or SMS);
- *Caller*: who makes the call or sends an SMS;
- *Callee*: who receives the call or SMS;
- *Street*: where the operation has taken place;
- *Time*: when the operation has taken place (ISO format[6] HH: MM: SS);
- *Duration*: how long the operation has been (ISO format HH: MM: SS);
- *Date*: when the operation has taken place (format: day, month, year).

The type of the operation is one of the following cases: "config", "gprs", "redirect", "out_sms(SUB_TYPE)", "in_sms(SUB_TYPE)", "out_call(SUB_TYPE)", "in_call(SUB_TYPE)". Sub-types are: "simple", "ack", "foreign".

The dataset has undergone the mandatory anonymization process for reasons of privacy and confidentiality. Therefore it does not contain data that allows tracing back to the real people involved in the investigative case. For instance, there is no phone number for the caller/callee but only a fictitious name. The names and the sizes (# rows) of the four files in the dataset are the following: Eudokia Makrembolitissa (8,783), Karen Cook McNally (20,894), Laila Lalami (12,689), and Lucy Delaney (8,480).

4.2 Preprocessing and ASP encoding of the dataset

The DigForASP dataset in its original format cannot be considered as a set of transactions in ASP syntax. It needs to undergo a transformation into the format described in Section 3. In short, each row of the dataset is encoded as a collection of facts through the *class* and *db* predicates. The transformation has been done by means of a Python script.

The classes refer to the operation type, namely: "in_sms", "out_sms", "in_call", "out_call", "config", "redirect", "gprs". The features are: *caller*, *callee*, *street_a*, *street_b*, *time*, *weekday* and *duration*. The *weekday* feature does not appear in the original dataset. It has been added with the following values: (0 = Monday,

[6] Format to describe dates and times: https://en.wikipedia.org/wiki/ISO_8601

..., 6 = Sunday). The *duration* feature has undergone a transformation in order to obtain a value expressed in seconds. The *time* feature has been discretized into four time slots: "morning" (from 06:00:00 to 11:59:59), "afternoon" (from 12:00:00 to 17:59:59), "evening" (from 18:00:00 to 23:59:59), and "night" (from 00:00:00 to 05:59:59). Depending on the analyst's needs, it is possible to consider (and encode) only the transactions related to specific days, months or years so as to subsequently carry out a more granular analysis. The transactions are sorted by date and time, as shown in Table 1.

Table 1: ASP encoding of some transactions from Karen's phone recordings from the morning of 07/09/2040 to the night of 08/09/2040.

07/09/2040 morning	07/09/2040 evening
`class(t1,in_sms).`	`class(t93,in_call).`
`db(t1,caller(lauretta_ngcobo)).`	`db(t93,caller(lady_anne_halkett)).`
`db(t1,callee(karen_cook_mcnally)).`	`db(t93,callee(karen_cook_mcnally)).`
`db(t1,street_a(bowsprit_avenue)).`	`db(t93,street_a(bigwood_court)).`
`db(t1,street_b(none)).`	`db(t93,street_b(none)).`
`db(t1,date(7,9,2040)).`	`db(t93,date(7,9,2040)).`
`db(t1,time(morning)).`	`db(t93,time(evening)).`
`db(t1,weekday(4)).`	`db(t93,weekday(4)).`
`db(t1,duration(0)).`	`db(t93,duration(56)).`

08/09/2040 night	
`class(t113,out_sms).`	`db(t113,caller(karen_cook_mcnally)).`
`db(t113,callee(karen_platt)).`	`db(t113,street_a(bayhampton_court)).`
`db(t113,street_b(none)).`	`db(t113,date(8,9,2040)).`
`db(t113,time(night)).`	`db(t113,weekday(5)). db(t113,duration(0)).`

4.3 Experiments

For the experiments here presented we have run the ASP encoding reported in Listing 1.1 over the largest file from the DigForASP dataset, namely Karen's phone records, made up of more than 20,000 rows. As regards the ASP solver, we have used the version 5.4.0 of *clingo*, with default solving parameters. The hardware and software platform used was a laptop computer with Windows 10 (with Ubuntu 20.04.4 subsystem), AMD Ryzen 5 3500U @ 2.10 GHz, 8GB RAM without using the multi-threading mode of clingo. Multi-threading reduces the mean runtime but introduces variance due to the random allocation of tasks. Such variance is inconvenient for interpreting results with repeated executions.

Exploratory tests During an investigation it is useful to understand what kind of information the extracted patterns can offer, in order to guide and support law enforcement in deciding the next steps to take during the investigation.

In Listing 1.2, as an illustrative example of the potential usefulness of contrast pattern mining in the DF field, we report the results obtained on Karen's phone records for the class "out_call". Here, we have set the minimum support threshold

to 10% and the maximum pattern length to 3. Overall, the nine contrast patterns returned by the algorithm provide a rich information about the habits of Karen as regards outgoing calls in contrast to other types of communication. Notably, they tell us that outgoing calls of Karen are mainly done in the morning (Line 8) or in the afternoon (Line 6). In particular, the answer at Line 4 highlights that outgoing calls are made mainly on Fridays.

```
1  in_pattern(caller(karen_cook_mcnally)) absolute_diff(430)
2  in_pattern(time(evening)) absolute_diff(24)
3  in_pattern(caller(karen_cook_mcnally)) in_pattern(time(
     evening)) absolute_diff(129)
4  in_pattern(weekday(4)) absolute_diff(14)
5  in_pattern(weekday(4)) in_pattern(caller(karen_cook_mcnally
     )) absolute_diff(126)
6  in_pattern(time(afternoon)) absolute_diff(34)
7  in_pattern(caller(karen_cook_mcnally)) in_pattern(time(
     afternoon)) absolute_diff(202)
8  in_pattern(time(morning)) absolute_diff(37)
9  in_pattern(time(morning)) in_pattern(caller(
     karen_cook_mcnally)) absolute_diff(103)
```

Listing 1.2: Contrast patterns for the "out_call" class.

Scalability tests With scalability tests, the goal is to assess the performance of the ASP encoding on datasets of increasing size. Once again, we have considered the file of Karen's phone records, from we have extracted 100, 1000 and 10,000 rows for the three groups of experiments. In each group, the experiments have been conducted by varying the class for the contrast and the minimum support threshold while keeping the maximum patterns length fixed to 3.

The first group of experiments considers the subset consisting of 100 rows. Observing Table 2, the class with the greatest contrast patterns concerns the "out_call" operation. With this order of magnitude, the extraction times of the patterns are less than one second for all classes. In general, the memory used for this operation is at most 25 MB.

The second group of experiments considers a subset consisting of 1,000 rows. From Table 3, we observe that the class with the greatest number of contrast patterns is again "out_call". It is worthwhile to note that, with an increase in the order of magnitude from hundreds to thousands, the execution time fluctuates in a range between 5 and 10 seconds with a minimum percentage variation equal to 400% (Figure 1 B). The memory consumed in this case is much higher than the previous batch of experiments since it jumps to a minimum of more than 300 MB, and a maximum that is around 460 MB (Figure 1 C).

The third group of experiments considers a subset consisting of 10,000 rows. Unlike the previous two groups, this group did not produce results because the amount of resources to be allocated to the RAM memory was so high (around

Table 2: Number of patterns, execution time (seconds), solver time (seconds) and memory consumption (MB) for 100 rows from Karen's phone records.

in_sms

Th.	#Pat.	Exec. t.	Solv. t.	Memory
10%	14	0.119	0.01	23.67
20%	0	0.087	0.00	22.11
30%	0	0.081	0.00	22.35
40%	0	0.089	0.00	22.11
50%	0	0.085	0.00	21.85

out_sms

Th.	#Pat.	Exec. t.	Solv. t.	Memory
10%	0	0.085	0.00	22.31
20%	0	0.076	0.00	21.93
30%	0	0.086	0.00	21.67
40%	0	0.086	0.00	22.31
50%	0	0.086	0.00	22.18

in_call

Th.	#Pat.	Exec. t.	Solv. t.	Memory
10%	21	0.137	0.03	24.22
20%	14	0.118	0.01	24.01
30%	7	0.120	0.01	24.01
40%	0	0.086	0.00	21.98
50%	0	0.084	0.00	21.44

out_call

Th.	#Pat.	Exec. t.	Solv. t.	Memory
10%	32	0.136	0.03	24.23
20%	14	0.122	0.02	24.23
30%	14	0.128	0.01	24.23
40%	7	0.121	0.01	24.23
50%	7	0.117	0.01	24.44

8GB) that the *clingo* process was killed by the operating system. Considering the pattern generation rule at Line 11 of Listing 1.1, the number of *item* atoms that must be combined to form the *in_pattern* atoms is equal to 2010. Instead, in the case of 100 and 1,000 rows the number of items are respectively 180 and 670. Since the total number of combinations is defined by

$$C_{n,k} = \binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{3}$$

and the minimum pattern length k varies from 1 to 3 in our tests, the total number of combinations for the problem in hand is given by the sum of:

- groupings of class 1: $\frac{2010!}{1!(2010-1)!}$;
- groupings of class 2: $\frac{2010!}{2!(2010-2)!}$;
- groupings of class 3: $\frac{2010!}{3!(2010-3)!}$.

It is clear that the computation required to solve the problem in hand is very heavy for a dataset size of tens of thousands rows or even more.

5 Final remarks

DPM is a promising direction of research in AI. We do not expect DPM to be competitive with dedicated algorithms, but to take advantage of the versatility of declarative frameworks to propose pattern mining tools that could exploit background knowledge during the mining process to extract less but meaningful patterns. Such tools are particularly welcome in application domains where

Table 3: Number of patterns, execution time (sec), solver time (sec) and memory consumption (MB) for 1,000 rows from Karen's phone records.

in_sms

Th.	#Pat.	Exec. t.	Solv. t.	Memory
10%	3	5.929	0.15	427.18
20%	0	5.178	0.00	345.7
30%	0	4.939	0.00	345.7
40%	0	4.843	0.00	345.7
50%	0	4.980	0.00	345.7

out_sms

Th.	#Pat.	Exec. t	Solv. t.	Memory
10%	0	4.979	0.00	336.36
20%	0	4.761	0.00	325.87
30%	0	4.715	0.00	336.36
40%	0	4.795	0.00	336.36
50%	0	4.733	0.00	323.02

in_call

Th.	#Pat.	Exec. t.	Solv. t.	Memory
10%	5	7.683	1.65	453.93
20%	1	6.834	0.71	453.92
30%	1	6.423	0.36	454.03
40%	0	4.916	0.00	346.46
50%	0	4.978	0.00	346.46

out_call

Th.	#Pat.	Exec. t.	Solv. t.	Memory
10%	9	10.155	3.87	465.07
20%	3	8.591	2.41	465.11
30%	1	7.603	1.40	464.89
40%	1	6.765	0.56	465.08
50%	0	4.945	0.00	354.01

the requirement of transparency is particularly crucial. This motivation is at the basis of a renewed interest of the AI community in declarative approaches. In particular, the expressive power of ASP makes the definition of algorithmic variants of the basic encoding pretty easy, mainly thanks to a clever use of constraints. Also, the availability of efficient ASP solvers encourage the use in applications characterized by combinatorial problems, such as the ones in pattern mining.

Contrast Pattern Mining is an interesting class of pattern mining problems. It is somehow halfway between discrimination and characterization of a data set, due to the use of class labels to guide the search for regularities. Nevertheless, to the best of our knowledge, it has not been addressed so far in DPM research. Our declarative approach is therefore a novel contribution to pattern mining which paves the way to new exciting AI applications. In particular, due to the inherent transparency, it appears to be suitable for analysing evidence in the context of DF investigations. As a case study we have considered the analysis of a real-world dataset of anonymised phone recordings. The preliminary results are encouraging, although they highlight some weaknesses. In particular, the combinatorial explosion affects the scalability of the approach. However, when compared to sequential pattern mining on the same dataset [15,16], it is noteworthy that in contrast pattern mining the solver takes much less time. This is partially due to the fact that the labeling of transactions with classes make the search space smaller.

For the future we plan to explore several directions of improvement of the work as regards efficiency and scalability. This implies different choices for the encoding, the solver, and the computing platform. Experiments could be, for instance, replicated with other ASP solvers, such as DLV2 [14], that revealed to

be scalable on large datasets. Hybrid ASP-approaches to pattern mining such as [18] could be adopted. An empirical evaluation of the approach with a more performant hardware is also planned. Besides the improvement of the current work, we intend to consider other variants of the contrast pattern mining problem. In parallel to the methodological work, we would like to benefit from a tighter interaction with DF experts in order to get their feedback as regards the validity and the usefulness of our work from DF viewpoint, and their suggestions for new interesting directions of applied research in this field.

Acknowledgments This article is based upon work from COST Action 17124 "Digital forensics: evidence analysis via intelligent systems and practices (DigForASP)", supported by COST (European Cooperation in Science and Technology). The work is also partially funded by the Università degli Studi di Bari "Aldo Moro" under the 2017-2018 grant "Metodi di Intelligenza Artificiale per l'Informatica Forense".

References

1. Brewka, G., Eiter, T., Truszczynski, M.: Answer set programming at a glance. Communications of the ACM **54**(12), 92–103 (2011). http://doi.acm.org/10.1145/2043174.2043195
2. Costantini, S., De Gasperis, G., Olivieri, R.: How answer set programming can help in digital forensic investigation. In: Ancona, D., Maratea, M., Mascardi, V. (eds.) Proceedings of the 30th Italian Conference on Computational Logic, Genova, Italy, July 1-3, 2015. CEUR Workshop Proceedings, vol. 1459, pp. 53–65. CEUR-WS.org (2015), http://ceur-ws.org/Vol-1459/paper29.pdf
3. Costantini, S., De Gasperis, G., Olivieri, R.: Digital forensics and investigations meet artificial intelligence. Ann. Math. Artif. Intell. **86**(1-3), 193–229 (2019). https://doi.org/10.1007/s10472-019-09632-y
4. Costantini, S., Lisi, F.A., Olivieri, R.: DigForASP: A European cooperation network for logic-based AI in digital forensics. In: Casagrande, A., Omodeo, E.G. (eds.) Proceedings of the 34th Italian Conference on Computational Logic, Trieste, Italy, June 19-21, 2019. CEUR Workshop Proceedings, vol. 2396, pp. 138–146. CEUR-WS.org (2019), http://ceur-ws.org/Vol-2396/paper34.pdf
5. De Raedt, L., Guns, T., Nijssen, S.: Constraint programming for data mining and machine learning. In: Twenty-Fourth AAAI Conference on Artificial Intelligence (2010)
6. Dong, G., Bailey, J.: Contrast data mining: concepts, algorithms, and applications. CRC Press (2012)
7. Gebser, M., Guyet, T., Quiniou, R., Romero, J., Schaub, T.: Knowledge-based sequence mining with ASP. In: IJCAI 2016-25th International joint conference on artificial intelligence. p. 8. AAAI (2016)
8. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Clingo = ASP + control: Preliminary report. arXiv preprint arXiv:1405.3694 (2014)
9. Gebser, M., Kaufmann, B., Kaminski, R., Ostrowski, M., Schaub, T., Schneider, M.: Potassco: The Potsdam answer set solving collection. AI Communications **24**(2), 107–124 (2011)
10. Guns, T., Dries, A., Nijssen, S., Tack, G., De Raedt, L.: MiningZinc: A declarative framework for constraint-based mining. Artificial Intelligence **244**, 6–29 (2017)

11. Guyet, T., Moinard, Y., Quiniou, R., Schaub, T.: Efficiency analysis of ASP encodings for sequential pattern mining tasks. In: Advances in Knowledge Discovery and Management, pp. 41–81. Springer (2018)
12. Han, J., Cheng, H., Xin, D., Yan, X.: Frequent pattern mining: current status and future directions. Data Min. Knowl. Discov. **15**(1), 55–86 (2007). https://doi.org/10.1007/s10618-006-0059-1
13. Jabbour, S., Sais, L., Salhi, Y.: Decomposition based SAT encodings for itemset mining problems. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. pp. 662–674. Springer (2015)
14. Leone, N., Allocca, C., Alviano, M., Calimeri, F., Civili, C., Costabile, R., Fiorentino, A., Fuscà, D., Germano, S., Laboccetta, G., Cuteri, B., Manna, M., Perri, S., Reale, K., Ricca, F., Veltri, P., Zangari, J.: Enhancing DLV for large-scale reasoning. In: Balduccini, M., Lierler, Y., Woltran, S. (eds.) Logic Programming and Nonmonotonic Reasoning - 15th International Conference, LPNMR 2019, Philadelphia, PA, USA, June 3-7, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11481, pp. 312–325. Springer (2019). https://doi.org/10.1007/978-3-030-20528-7_23
15. Lisi, F.A., Sterlicchio, G.: Declarative pattern mining in digital forensics: Preliminary results. In: Calegari, R., Ciatto, G., Omicini, A. (eds.) Proceedings of the 37th Italian Conference on Computational Logic, Bologna, Italy, June 29 - July 1, 2022. CEUR Workshop Proceedings, vol. 3204, pp. 232–246. CEUR-WS.org (2022), http://ceur-ws.org/Vol-3204/paper_23.pdf
16. Lisi, F.A., Sterlicchio, G.: Mining sequences in phone recordings with answer set programming. In: Bruno, P., Calimeri, F., Cauteruccio, F., Maratea, M., Terracina, G., Vallati, M. (eds.) HYDRA - RCRA 2022: 1st International Workshop on HYbrid Models for Coupling Deductive and Inductive ReAsoning and 29th RCRA workshop on Experimental evaluation of algorithms for solving problems with combinatorial explosion. CEUR Workshop Proceedings, vol. ?, pp. ?–? CEUR-WS.org (2022)
17. Negrevergne, B., Guns, T.: Constraint-based sequence mining using constraint programming. In: International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research. pp. 288–305. Springer (2015)
18. Paramonov, S., Stepanova, D., Miettinen, P.: Hybrid ASP-based approach to pattern mining. Theory Pract. Log. Program. **19**(4), 505–535 (2019). https://doi.org/10.1017/S1471068418000467

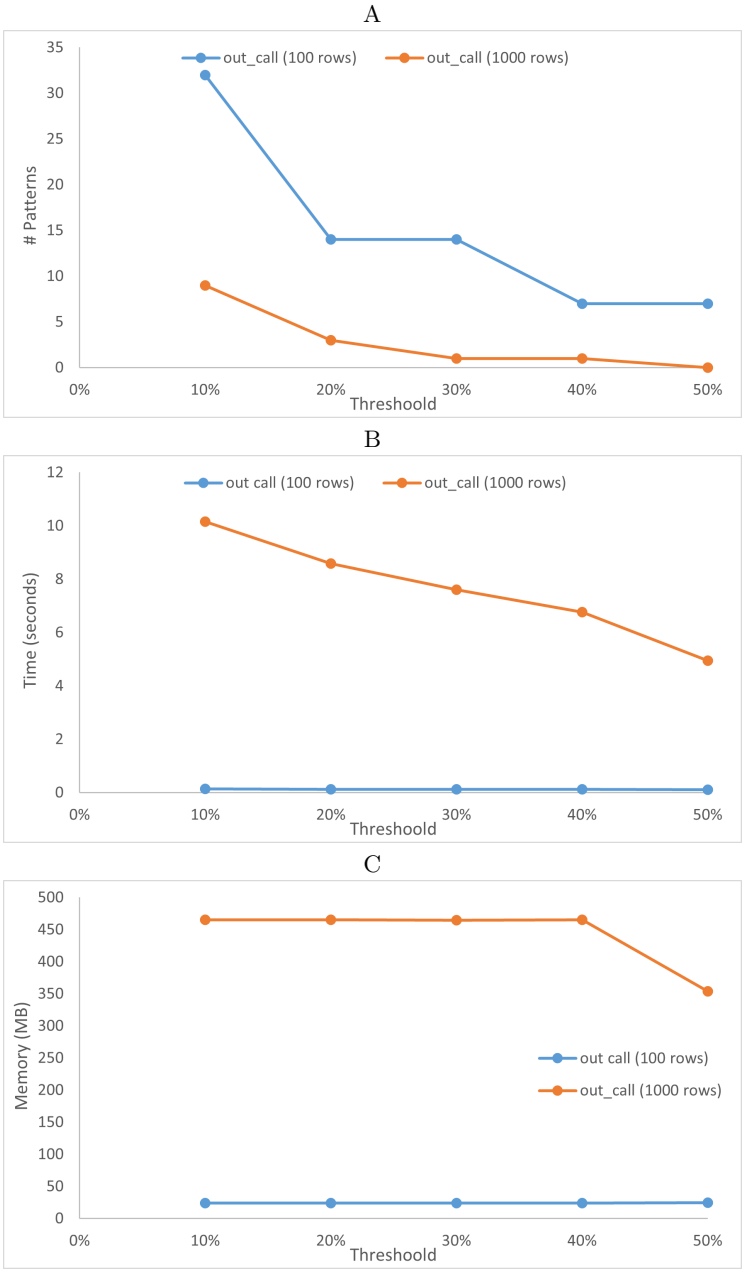

Fig. 1: Comparison w.r.t. the number of patterns extracted (A), execution time (B) and memory consumption (C) for the "out_call" class (Tables 2 and 3).