A Latitudinal Study on the Use of Sequential and Concurrency Patterns in Deviance Mining

Laura Genga, Domenico Potena, Andrea Chiorrini, Claudia Diamantini, and Nicola Zannone

Abstract Deviance mining is an emerging area in the field of Process Mining, with the aim of explaining the differences between normal and deviant process executions. Deviance mining approaches typically extract representative subprocesses characterizing normal/deviant behaviors from an event log and use these subprocesses as features for classification. Existing approaches mainly differ for the employed feature extraction technique and, in particular, for the representation of the patterns extracted, ranging from patterns consisting of sequence of activities to patterns explicitly representing concurrency. In this work, we perform a latitudinal study on the use of sequential and concurrency patterns in deviance mining. Comparisons between sequential and concurrency patterns is performed through experiments on two real-world event logs, by varying both classification and feature extraction algorithms. Our results show that the pattern representation has limited impact on classification performance, while the use of concurrency patterns provides more meaningful insights on deviant behavior.

Laura Genga
Eindhoven University of Technology, e-mail: l.genga@tue.nl

Domenico Potena
Università Politecnica delle Marche, e-mail: d.potena@univpm.it

Andrea Chiorrini
Università Politecnica delle Marche, e-mail: a.chiorrini@univpm.it

Claudia Diamantini
Università Politecnica delle Marche, e-mail: c.diamantini.it

Nicola Zannone
Eindhoven University of Technology, e-mail: n.zannone@tue.nl

1 Introduction

Process Mining (PM) comprises a family of analysis techniques that aim to gain valuable insights into processes from raw data recording process executions [1]. PM techniques take as input an *event log* in which past process executions are recorded as *traces*, i.e. sequences of activities performed within a process execution. By analyzing these traces, PM techniques allow understanding how processes are actually performed within an organization, providing valuable insights for both the enhancement of existing models and the management of processes for which accurate models do not exist.

In this work, we focus on a particular class of PM techniques, namely *deviance mining* techniques. These techniques aim to explain the differences between "normal" and "deviant" process executions [9, 25], typically in terms of *control-flow*. Accordingly, deviance mining approaches usually extract from an event log *subprocesses* (i.e., patterns of activities) representing work practices characterizing desired/undesired behaviors and use these subprocesses as features for classification. The notion of "deviant" depends on the context and purposes of the analysis. Here, we assume that *deviant* executions have been marked as such by an analyst or by means of an automatic labeling system based, for instance, on some process constraints or process performance metrics.

A variety of approaches for deviance mining have been proposed in the last years, leveraging different techniques for feature extraction and classification. A main difference between these approaches is the type of features extracted for classification. While a number of proposals like [3–5, 8, 9, 22, 26, 27] extract subprocesses in form of sequence of activities (hereafter called *sequential patterns*), few work [12, 17, 21] has proposed the use of more representative subprocesses that explicitly represent the presence of concurrent activities in the process (hereafter called *concurrency patterns*). Despite the pattern representation could have a significant impact on the quality of deviant mining techniques, to date no studies have evaluated the effect of these representations on the deviance mining problem.

In this work, we investigate the advantages and disadvantages of the use of sequential vs. concurrency patterns in deviance mining. In particular, we focus on the discriminative power of features and their capability to provide a better understanding of deviant behaviors. Comparisons between sequential and concurrency patterns is performed through experiments on two real-world event logs, by varying both classification and feature extraction algorithms. Results have been discussed also analyzing the main characteristics of extracted patterns. According to our results, neither of the two pattern representations prevails the other in terms of classification performance. The situation, however, is different with respect to the understandability of deviant behavior. Our analysis shows that the use of concurrency patterns provides analysts with more meaningful insights on the process.

The remainder of the paper is organized as follows. The next section presents an overview of deviance mining and discusses existing approaches. Section 3 discusses the challenges and benefits of using sequential patterns and concurrency patterns. Section 4 presents an experimental evaluation and Section 5 discusses lessons learned. Finally, Section 6 concludes the paper and provides directions for future work.

2 Deviance mining

Deviance mining encompasses two main types of analysis. The first is *log delta analysis* [2,27], which is mainly descriptive and aims to infer two different models representing normal and deviant traces and compare these models to identify possible differences. The other type of analysis is *sequence classification*. This is a predictive task whose goal is to automatically classify log traces as normal or deviant, adopting a classification model learned from past process executions. In the following, we mainly focus on the latter since it is more close to our work.

Sequence classification approaches usually encompass three main steps: *(i) feature extraction*, *(ii) feature selection* and *(iii) classification*. During feature extraction, subprocess mining techniques are applied to extract a set of *representative* patterns (i.e., subprocesses, possibly consisting of only one activity) from an event log. These patterns are considered as candidate features for classification. Since subprocess mining techniques might return a huge set of subprocesses, feature selection techniques are often applied to identify *relevant* features. The core idea is to remove all those features that either do not have an impact on the classification or are "redundant", i.e. they are correlated to other features or can be obtained as a combination of other features [15, 24]. In the final step, relevant features are used to train a classifier and discover characteristics of deviant traces.

Existing approaches mainly differ for the adopted pattern extraction technique. Nguyen et al. [25] survey the most common types of features used by sequence classification algorithms, grouping them in: i) *activity-based* patterns, where each feature corresponds to an activity of the process [27]; ii) *sequence-based* patterns, which capture recurrent executions of sequences of activities, adopting classic sequence pattern mining algorithms [8] or ad-hoc algorithms looking for sequences representing typical process control-flow structures, e.g. loops and subprocesses [3, 4]; iii) *discriminative* patterns, which are sequence-based patterns where, however, features are chosen also by taking into account their discriminative power with respect to deviant/normal traces [22, 26]. Some approaches propose to mix pattern families, like in [5]. However, this can result in a sparse and cumbersome representation of log traces, which can easily lead to problems known as the curse of dimensionality [16]. To address this issue, Cuzzocrea et al. [9] propose an ensemble-learning approach, where first multiple learners are trained on different families of patterns and then a final model is obtained by means of a stacking procedure, leveraging the predictions of the base models.

The aforementioned approaches only consider *sequential* patterns, i.e. either single activities or sequences of activities ordered according to their temporal execution, as classification features, thus overlooking the control-flow structure of process executions. An exception is represented by the approach in [3, 4]. Here, the authors deal to some extent with concurrency by introducing the so-called *alphabet patterns*, which are groups of sequences of activities sharing the same names. However, this only provides an approximation (often, imprecise) of concurrent behaviors and, anyway, does not show the structure of concurrency.

In this respect, a few subprocesses mining approaches propose to extract *concurrency* patterns from sequential log traces [17, 21, 28]. For instance, Leemans et al. [21] propose

an approach to derive "episodes", i.e. directed graphs where nodes correspond to activities and edges to *eventually-follow* precedence relations. Therefore, an edge between two activities means that one occurs before the other, but other activities may occur in between. Therefore, multiple episodes can be returned for the same set of activities. Hwang et al. [17] derive "temporal graphs", where ordering relations are defined exploiting starting and completion time of events. This assumption, however, constrains the applicability of the approach to event logs storing such attributes.

Genga et al. [12] propose a more general approach for deviance mining based on the conversion of sequential log traces into *Instance Graphs* (IG), which are directed graphs showing the control-flow of each process execution. Roughly speaking, the idea underlying the building of an IG is to derive (when not known a priori) the ordering relations among process activities and build for each trace a graph whose nodes correspond to process activities, connected according to the (derived) ordering relations. Different strategies have been proposed to build IGs from sequential log traces (see [10] for an overview). The main differences between these techniques lie in the different assumptions on the event log and/or on the process model (e.g., the presence of special attributes or a high degree of structure in the process). Therefore, the selection of the most suitable technique depends on whether the process and the event log at hand fit the assumptions of the preferred technique.

The conversion of log traces into IGs allows the use of subgraph mining algorithms to infer the set of representative subprocesses (see [18] for an overview of well-known subgraph mining techniques). For instance, the work in [12] employs SUBDUE [19], an iterative compression-based subgraph mining algorithm that evaluates the relevance of a subgraph in terms of its *description length*, i.e. the number of bits needed to encode its nodes and edges. Another well-known subgraph mining algorithm is Frequent Subgraphs Discovery (FSG) [20]. Starting from single nodes, FSG iteratively builds a set of *subgraphs candidates* by adding one edge to the previous set of candidates and discards those with a support lower than a user-defined threshold. FSG *Max* is a popular variant of FSG, which returns only the *maximal subgraphs*, i.e. those subgraphs that are not involved in any other [18]. Greco et al. [14] propose a subgraph mining algorithm that exploits knowledge about relationships among activities (e.g., AND/OR splits) to drive subgraphs mining. Graphs are generated by replaying traces over the process model; however, this algorithm requires a model properly representing the event log, which may not be available for many real-world processes.

The application of pattern extraction techniques to an event log can result in a large number of features, which might not be all relevant for classification. Feature selection methods have been proposed to prune a feature set by identifying those features relevant for classification (see [7] for a survey). Existing methods are typically grouped in three main categories. *Filter* methods evaluate the correlation between each feature and the class, selecting only those features whose value is above a user-defined threshold. *Wrapping* methods adopt an optimization strategy; given a classifier, they look for the subset of features that provide the best results in terms of accuracy. The search is performed either by adding or removing one feature at each iteration and, then, evaluating the improvements led by such a feature. Finally, *embedded* methods are feature selection approaches embedded in the classification process as, e.g., in the case of decision trees.

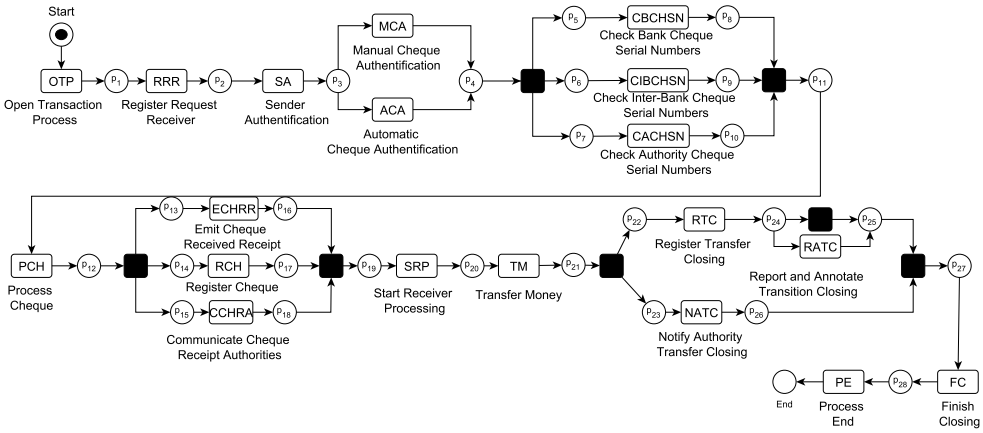


Fig. 1: Money transfer process represented using Petri net notation

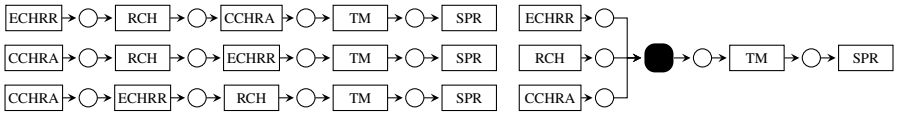
3 Sequential vs. Concurrency: challenges and benefits

This section is devoted to highlight the main differences between the use of sequential and concurrency patterns in deviance mining. To make the discussion more concrete, we use a money transfer process as an illustrative example. Fig. 1 shows a simple model for this process, represented using Petri net notation.[1] Boxes, circles and black boxes represent transitions, places and invisible transitions respectively. The text below each transition is the activity label, and its abbreviation is shown inside the box.

The process starts by registering the opening of a transaction and the registration of the request (`OTP` and `RRR`) along with the authentication of the person requesting the money transfer (`SA`). Then, a preliminary cheque verification is performed, either manually (`MCA`) or automatically (`ACA`), followed by further control steps in which the serial of the cheque is searched within three databases: the internal database of the bank (`CBCHSN`), the database of the consortium the bank belongs to (`CIBCHSN`) and the database of an international authority (`CACHSN`). Then, the cheque is processed by registering it (`RCH`), issuing a receipt (`ECHRR`) and reporting the transfer to authorities (`CCHRA`). Subsequently, the receiver is checked (`SRP`) and the money is transferred (`TM`). At the end of the process, we have the registration of the transfer closing (`RTC`), possibly with additional annotations (`RATC`), the notification to the authorities of the closing (`NATC`) and the final registration of the closed transfer (activities `FC` and `PE`). Let us suppose that a fraudulent behavior occurred in some executions, which led to an illegal transfer of money, for instance, the receiver of the transfer was checked (`SRP`) after transferring the money (`TM`). This behavior have been labeled as "deviant" by the bank.

By applying deviance mining, we would like to extract patterns able to distinguish "good" from "bad" traces. The adopted pattern representation, however, has a significant impact on the patterns extracted. For instance, if we only extract patterns consisting

[1] This process is a simplified and revised version of the process described in [6].



(a) Sequential patterns (b) Concurrency pattern

Fig. 2: Deviant patterns

of sequences of activities, the deviant behavior above would be captured by the three sequential patterns in Fig. 2a. On the other hand, using patterns able to account for concurrency, we can capture the deviant behavior with a single concurrency pattern as illustrated in Fig. 2b. We expect that the choice of the pattern representation (sequential or concurrency) has an impact on the deviance mining problem.

It is well known that the quality of a classification model strongly depends on the features used for classification. Indeed, greater the discriminative power of the features used, better the classifier's performances are. Although any concurrency pattern can be represented as a set of sequential patterns, thus providing similar classification performance, there are cases where using sequential patterns or using concurrency patterns can result in different performances. For instance, sequential patterns typically have an overall support (i.e., the percentage of process instances in which the pattern occurs) much lower than the corresponding concurrency pattern. Thus, they can be discarded during feature extraction. As a consequence, some deviant behavior would not be used for classification, thus resulting in poor performances. This is particularly challenging in deviance mining as the deviant class is typically underrepresented in the dataset and, thus, patterns representing deviant behavior are not expected to have a high support. In contrast, if the deviant behavior occurs in only some of the 'branches' of the concurrency pattern in Fig. 2b, i.e. the pattern describes both normal and deviant behavior, the use of sequential patterns in Fig. 2a would provide better performances as they allow for a more fine-grained representation of behaviors.

The choice of the pattern representation has also a significant impact on the understanding of deviant behavior. Sequential patterns only consider the temporal ordering of activities, thus hiding the underlying control-flow of the process and providing the analyst with an inexact representation of the process. It may not be trivial for an analyst to recognize concurrent behavior from sequential patterns even in the simple example of Fig. 2. On the other hand, concurrency patterns provides the analyst with a more comprehensible and accurate description of deviant behavior.

Based on the observations above, we hypothesize that the pattern representation has an impact on both classification performance and understendability of deviant behavior. In particular, we are interested in two main research questions:

(R1) *Do concurrency patterns have more discriminative power compared to sequential patterns?*

(R2) *Does the use of concurrency patterns provide a better understanding of deviant behaviors compared to the use of sequential patterns?*

To verify our hypotheses, in this work we perform a latitudinal study on the use of sequential patterns and concurrency patterns. In particular, we apply a variety of deviance mining approaches (obtained by assembling various techniques for feature extraction

and classification) on two real-life datasets. In the next section we present the setting and results of our experiments and in Section 5 we provide an in-depth analysis of the results.

4 Experiments

This section presents our experiments on two real-world datasets, with and without accounting for possible concurrency among process activities. We first present the experimental settings and the main characteristics of the datasets and, then, report the results. The experiments were performed in RapidMiner.[2] The RapidMiner processes, scripts and results are available at `http://daisy.dii.univpm.it/NFmcp/`.

4.1 Datasets

For our experiments, we used two real-world event logs, namely the *BPI2012* and *Sepsi* logs (described below). Table 1 shows statistics of these logs, including the number of traces, activities and events, along with the minimum, maximum and mean number of events in each trace. For each log, we created two datasets, one in which log traces are converted into sequential IGs (i.e., IGs consisting of sequences of activities) and one in which log traces are converted into IGs exhibiting concurrency.

BPI2012: This event log records the loan management process of a Dutch Financial Institute and was made available for the 2012 BPI challenge.[3] The log contains the events recorded for three intertwined subprocesses, one describing how loan applications should be handled, one describing the handling of loan offers, and one specifying how work items are processed. This event log has been largely studied in the literature. Here, we rely on the analysis performed in [13], which identified a set of *deviant* behaviors based on the temporal duration of the handling of an application. More precisely, the authors pointed out that, although approved applications were typically completed within 27 days, a number of approved applications required much more time, up to five months. Moreover, a small set of applications were approved within a day from their request. Elaborating upon this, in our study we marked applications approved within 27 days as *normal* and applications whose approval required a longer times as *deviant*. We filter out from the log those applications approved within one day, since they represent a different kind of deviant behavior.

Sepsi: This event log tracks the trajectories of 1050 patients with symptoms of a sepsis condition and comprises events on activities in the emergency room, admission to hospital wards, and discharge, as well as data from laboratory tests and triage checklists.[4] Following the analysis in [23], we mark traces concerning patients that had to return to the emergency room within a month from their discharge as deviant, and traces concerning

[2] `https://rapidminer.com/`

[3] The log is available at `https://www.win.tue.nl/bpi/doku.php?id=2012:challenge`.

[4] The log is available at `https://data.4tu.nl/repository`.

Dataset	Cases	Normal	Deviant	Activities	Events	Min Events	Max Events	Mean Events
BPI2012	2169	270	1899	21	97 343	22	163	45
Sepsi	1050	294	756	16	15 214	3	185	14

Table 1: Statistics of the BPI2012 and Sepsi logs.

patients for which the treatment seemed to be effective (i.e., they did not have to come back) as normal.

4.2 Settings

The application of deviance mining to an event log requires selecting suitable techniques for: *(i)* feature extraction, *(ii)* feature selection and *(iii)* classification. Moreover, we need metrics for comparing the performance of the methods under examination. In this section, we describe the techniques used for each phase in our experiments and evaluation metrics.

Feature extraction: In our experiments, we consider three widely-used subgraph mining techniques for feature extraction, namely SUBDUE, FSG and FSG Max. We used the implementation of SUBDUE provided at `http://ailab.wsu.edu/subdue/` (in its default configuration) and the implementation of FSG and FSG Max provided at `http://glaros.dtc.umn.edu/gkhome/`. For the latter, we varied the support threshold in the interval [0.5,0.7], with a step of 0.1.

As discussed in Section 2, the use of subgraph mining techniques requires converting log traces in IGs. We used the technique presented in [10] to generate the dataset comprising IGs exhibiting concurrency. To generate the dataset comprising sequential IGs, we converted each sequential trace into a graph by transforming each event in a node and adding an edge between each pair of consecutive nodes. The two datasets were given as input to the subgraph mining techniques in order to derive the set of representative subgraphs, which represent our feature set. Hereafter, we refer to the feature set extracted from sequential IGs as the *sequential* feature set and to the feature set extracted from IGs exhibiting concurrency as the *concurrency* feature set.

Feature selection: We exploited an embedded method to select the set of most relevant features as it provides a trade-off between the need of accounting for possible feature interactions in evaluating their relevance and computational efficiency. Specifically, we built a decision tree from the occurrence matrix of each log and, then, kept only those features occurring in the obtained model. For our experiments, we used the implementation of the decision tree classifier provided by RapidMiner, by setting a maximum depth of the tree to 50.

Classification: We employed four well-known and widely used classifiers, i.e. *Support Vector Machine* (SVM), with a Gaussian radial basis function kernel, *K-Nearest Neighbour* (K-NN), *Decision Tree* (DT) and *Random Forest* (RF), for the classification step. To select the main classification parameters for each algorithm, we built a grid optimization

Algorithm	Parameter	Min	Max	N_Steps	Scale	Value set
SVM	γ	10^{-6}	1	7	log	[10^{-6},10^{-5},10^{-4},10^{-3},10^{-2},10^{-1},1]
	C	1	10^5	6	log	[1,10,10^2,10^3,10^4,10^5]
K-NN	k	2	200	4	log	[2,6,20,63,200]
DT	depth	2	30	3	log	[2,5,12,30]
RF	n. trees	3	100	4	linear	[3,27,52,76,100]
	depth	2	30	3	log	[2,5,12,30]

Table 2: Intervals tested for deriving the optimal combination of classification parameters

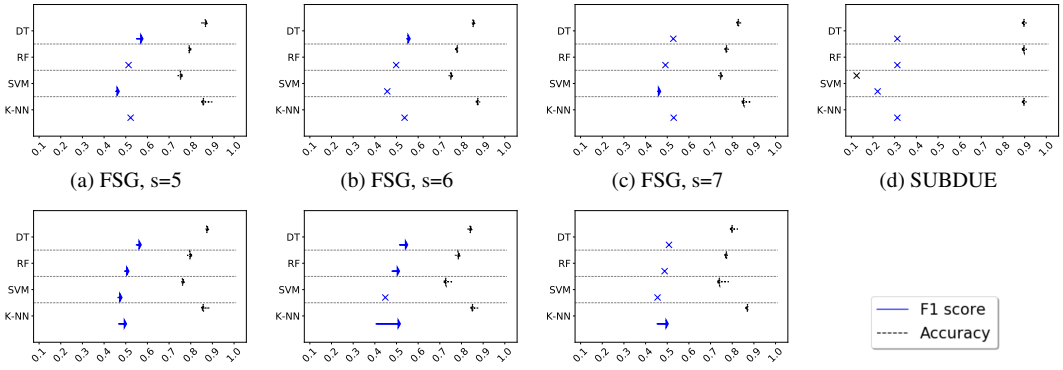
process; namely, we performed an extensive experimentation over multiple combinations of the parameters, to derive the combination returning the best results in terms of the selected performance metric. We optimized the following set of parameters for each classifier: the γ and C parameters for SVM, which, respectively, are used to determine the variance of the Gaussian kernel and the impact of each support vector on the boundaries of the feature space partition; the number of clusters k for K-NN; the *maximum tree depth* for DT; and, finally, the maximum *number of trees* and the *maximum tree depth* for RF. Table 2 shows the configurations adopted for the optimization process for each classifier. Intervals and steps were set experimentally, with the aim to explore a large range of values for the selected parameters while keeping reasonable computation time.

Since the datasets used in the experiments are highly unbalanced (see Table 1), we preprocess the datasets using the *MetaCost* algorithm before performing classification. MetaCost [11] is a well-known resampling algorithm to modify the class distributions, reducing the imbalance of a dataset. It is based on wrapping a meta-learning stage around an error-based classifier in such a way that the classifier effectively minimizes costs while seeking to minimize zero-one loss. We used the implementation of MetaCost provided by RapidMiner, using 100 iterations and 80% of the dataset for resampling.

Evaluation Metrics: We evaluated the tested methods along two dimensions: *classification performance* and *understandability* of deviant behavior. The first dimension is measured in terms of two standard performance measures, namely the *accuracy* of the classifier and *F1 score* (for the deviant class), which is the harmonic average of precision and recall. For the second dimension, we considered the *average feature size*, defined as the average number of nodes in the patterns used for classification. The reason underlying the latter measure is that larger patterns represent larger portions of process behaviors and, hence, provide the analyst with more meaningful information.

4.3 Results

Classification performance: Fig. 3 and Fig. 4 show the variation of classification performance when moving from the sequential to the concurrency feature set on the BPI2012 and Sepsi logs, respectively. Each plot presents the results for each tested subgraph mining configuration. The x-axis represents the accuracy and F1 obtained



(a) FSG, s=5 (b) FSG, s=6 (c) FSG, s=7 (d) SUBDUE

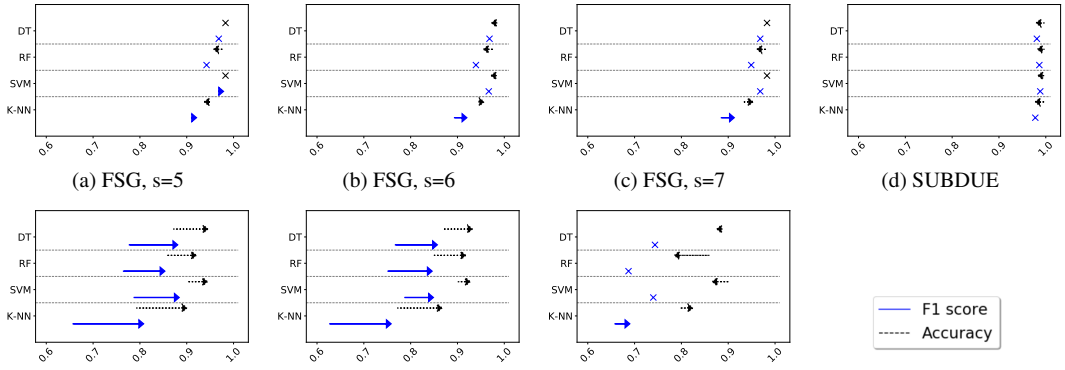
(e) FSG Max, s=5 (f) FSG Max, s=6 (g) FSG Max, s=7 (h) legend

Fig. 3: Classification performance for the BPI2012 log

by the classifiers in that configuration whereas the y-axis reports the tested classifiers. In correspondence to each classifier, we draw two arrows representing the change in F1 score (solid blue arrow) and accuracy (dotted black arrow) when moving from the sequential to the concurrency feature set. Arrows directed from the left to the right represent an improvement of performance when using the concurrency feature set; and the other way around. Crosses are used when we obtained exactly the same results in terms of accuracy (black cross) or F1 (blue cross).

The figures show that the sequential and the concurrency feature sets provide similarly results in terms of classification performance, where the differences are of the order 10^{-1}. Although the concurrency feature set, in general, does not provide significant improvements in performance, there are a few cases where the improvement is notable. For example, we can observe an improvement between 5% and 10% when using FSG Max with support of 5 and 6 in Sepsi. Overall, these results do not show a clear advantage in the use of the sequential or the concurrency feature set on both datasets.

Understandability: Table 3a and Table 3b report the average size of patterns in the sequential and concurrency feature sets obtained using various subgraph mining algorithm configurations on the BPI2012 and Sepsis logs, respectively. We can observe that patterns extracted from IGs exhibiting concurrency tend to be larger than the one extracted from sequential IGs. On average, patterns in the concurrency feature set comprise at least two nodes more than patterns in the sequential feature set, for both event logs. This difference becomes even more evident when considering FSG Max, where we observe an increasing of at least five nodes, on average, in the BPI2012 log and of at least six in the Sepsi log. Also when considering SUBDUE, we can observe a significant increase in the pattern size in the Sepsis log (from patterns consisting of two nodes to patterns with more than twelve nodes). Overall, the patterns in the concurrency feature set turned out to be significantly larger than the patterns in the sequential feature set for all tested configurations.



(a) FSG, s=5 (b) FSG, s=6 (c) FSG, s=7 (d) SUBDUE

(e) FSG Max, s=5 (f) FSG Max, s=6 (g) FSG Max, s=7 (h) legend

Fig. 4: Classification performance for the Sepsis log

Alg	Par	Avg size seq	Avg size conc
FSG	s=5	6.602	8.839
	s=6	6.597	8.503
	s=7	6.144	8.378
FSG Max	s=5	12.078	17.197
	s=6	10.987	16.130
	s=7	10.875	15.950
SUBDUE	n.a.	33.735	37.490

(a) BPI2012

Alg	Par	Avg size seq	Avg size conc
FSG	s=5	2.89	5.27
	s=6	2.68	5.72
	s=7	2.91	4.73
FSG Max	s=5	4.25	9.27
	s=6	3.95	9.67
	s=7	3.91	10.21
SUBDUE	n.a.	2.00	8.14

(b) Sepsis

Table 3: Average size of feature sets for the BPI2012 and Sepsis logs

5 Discussion

Our experiments show that neither the sequential nor the concurrency feature set can be considered to be the absolute 'winner' in terms of classification performance in none of the tested datasets. To delve into the causes of these results, we analyzed the main characteristics of the two feature sets.

The most likely explanation for this outcome is that the two feature sets share some of the most relevant features for classification, thus leading to similar performance. In other words, the patterns with the strongest discriminative power for identifying deviant traces are actually sequential. To verify this hypothesis, we report in Table 4 some characteristics of both feature sets. For both sequential and concurrency feature sets, we report the number of extracted patterns (**Ext.**) and the number of patterns selected for classification (**Sel.**). In addition, for the concurrency feature set, we report the number of patterns actually exhibiting concurrency (**TC**).[5] We also report the number of patterns in common between the two feature sets, both as absolute number (**Num**) and in terms of the percentage of the shared patterns w.r.t. each set (**%_Conc** and **%_Seq**).

[5] Note that the concurrency feature set comprises both sequential and concurrency patterns.

Dataset	Alg	Par	Sequential		Concurrency			Shared		
			Ext.	Sel.	Ext.	Sel.	TC	Num	%_Conc	%_Seq
BPI2012	FSG	s=5	1974	151	133 541	168	37	49	29.17	32.45
		s=6	1657	134	99 456	155	29	52	33.55	38.81
		s=7	1383	125	72 154	143	30	45	31.47	36.00
	FSG Max	s=5	100	90	252	122	77	14	11.48	15.56
		s=6	90	79	220	108	68	16	14.81	20.25
		s=7	79	72	180	100	60	17	17.00	23.61
	SUBDUE	n.a.	4340	49	2887	49	37	1	2.04	2.04
Sepsis	FSG	s=5	194	28	37 249	26	7	1	3.85	3.57
		s=6	144	25	29 584	25	9	4	16.00	16.00
		s=7	125	23	21 233	26	6	5	19.23	21.74
	FSG Max	s=5	51	51	399	67	57	0	0.00	0.00
		s=6	40	39	483	67	60	0	0.00	0.00
		s=7	34	34	367	57	53	1	1.75	2.94
	SUBDUE	n.a.	1822	5	739	14	7	0	0.00	0.00

Table 4: Properties of patterns extracted from the BPI2012 and Sepsi logs

We can observe that there is a large overlap between the two feature sets when the FSG algorithm was applied to the BPI2012 log, where the percentage of share features ranges between 30% and 40%. This percentage decreases when considering FSG Max, where, however, it is still around 10-15%, and becomes almost 0 when considering SUBDUE. Also for Sepsi, we can note that the highest percentage of shared features are in the sets extracted by FSG (although with values significantly lower than the ones for the BPI2012 log); while, for FSG Max and SUBDUE, we obtained almost disjoint sets.

Comparing the results in Table 4 with the ones in Figs. 3 and 4, we can note that a large number of shared features often leads to similar classification performance, especially concerning the F1 score. However, similar performance is also obtained with different feature sets. For example, Fig. 4a shows small differences in terms of both accuracy and F1 score for most of the classifiers, with the exception of RF. In particular, using DT we obtained exactly the same results for both measures However, the two feature sets have few elements in common, indicating that the two feature sets encompass different patterns but with similar discriminative power.

The impact of a chosen feature set on the classification performance largely depends on the dataset at hand. As discussed in Section 3, accounting for concurrency likely leads to better classification performance when the deviant behavior involves concurrency. Otherwise, the use of concurrency might not have any appreciable impact or even lead to worse performance, for instance, when sequential patterns can properly model the deviant behavior. This is the case, for instance, for the BPI2012 and Sepsis logs. As shown in Table 4, many patterns in the concurrency feature set are sequential patterns. This indicates that, in those logs, deviant behavior can be largely captured by sequential patterns.

Since the extraction of concurrency patterns has a higher computational cost, being the conversion of sequential traces into IGs more complex and time consuming when concurrency is taken into account, we can conclude that the use of the sequential feature set is, in general, more convenient, unless there is some a-priori knowledge on the process suggesting that deviant behavior involves concurrency.

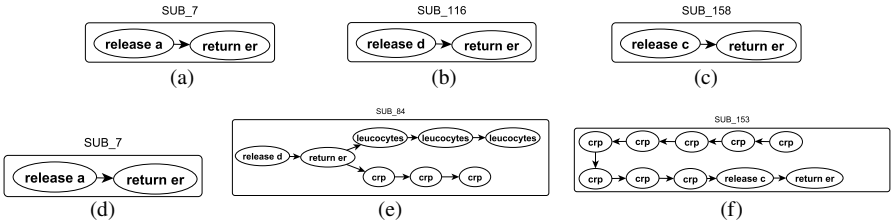


Fig. 5: First three features, in order of relevance, selected over the sequential (a,b,c) and the concurrency (d,e,f) feature sets extracted by SUDBUE on the Sepsis log.

The situation, however, is different with respect to the understandability of deviant behavior. Table 3 clearly shows that the concurrency feature set outperform the sequential feature set with respect to the average pattern size in all tested configurations. This implies that, by accounting for concurrency, we were able to obtain significantly larger portions of the process behavior, which are likely to provide more information and more meaningful insights on the process and, hence, can better support the analyst in understanding the root causes of deviant behavior. As discussed above, such an improvement is obtained with little or no loss in terms of classification performance.

As an example, Fig. 5 illustrates the first three more relevant features of the sequential (Fig. 5a to Fig. 5c) and the concurrency (Fig. 5d to Fig. 5f) feature sets obtained by applying SUBDUE to the Sepsis log. We can note that both sets share the first feature, which is the most discriminative. This confirms that, for this log, deviant behaviors are best described by means of sequential patterns. The behavior captured by the other two patterns in the sequential feature set are also captured by the patterns in the concurrency feature set. However, they are wrapped in a larger behavior. It is easy to observe that the patterns in Figs. 5e and 5f provide more meaningful insights on the process behavior than the corresponding patterns in the sequential feature set (Figs. 5b and 5c), by also highlighting the context in which the deviant behavior occurs. Recall from Section 4.1 that deviant traces in the Sepsis log correspond to patients who had to return to the emergency room within one month from the treatment. Therefore, it comes with no surprise that activity '`return er`' occurs in discriminant features. However, an analyst may also be interested in the possible causes that could justify the return of the patient to the emergency room; namely, he is interested in identifying commonalities between the paths of patients in the deviant traces, in addition to predict their return to the emergency room. In this respect, the patterns in the sequential feature set only allow us to deduce that three out of four release protocols were related to episodes of relapse. The patterns in the concurrency feature set, instead, show us larger portions of patients' trajectories, reporting also the typical set of exams that were performed for those patients before their release, which is a potentially useful information for better understanding what happened.

Summing up, the choice of the pattern representation (sequential vs. concurrency) in deviance mining largely depends on analysts' needs and dataset at hand. When the building of efficient classifiers is the main target, the use of the sequential feature set is likely the best option, since the mining process is computational less-expensive; instead,

when the main goal is delving into the root causes of deviant behaviors, accounting for concurrency usually leads to larger and more meaningful patterns.

6 Conclusion

In this work, we investigated the impact of accounting for concurrency in deviance mining by experimentally assessing advantages and drawbacks of using sequential and concurrency patterns along two main dimensions: *(i)* discriminative powers of the feature set and *(ii)* the extent to which they support an human analyst in understanding deviant behavior.

We performed a number of experiments on two real-world event logs, comparing the results obtained by four well-known classifiers on the sequential and the concurrency feature set. The obtained results show that the two feature sets provide similar results in terms of classification performance. However, patterns obtained when accounting for concurrency are on average significantly larger than the patterns in the sequential feature set. As a consequence, whether accounting for concurrency or not mainly depends on the analyst's goals. If the main focus is to build an efficient classifier to label as deviant/normal future process executions, then sequential patterns are likely to be a better choice. In fact, extracting the concurrency feature set has higher computational costs, without significant classification performance benefit. Instead, when the analysis is aimed at understanding the possible causes of deviant behavior, the concurrency feature set is preferable as patterns are potentially more meaningful for a human analyst and can better support her in delving into the causes of deviant behavior.

In future work, we plan to extend the experimental set in order to delve more into the differences between sequential and concurrency patterns. First, we intend to explore the impact of concurrency with respect to different feature selection techniques, as well as with respect to different strategies for dealing with unbalanced datasets. Moreover, we intend to investigate other techniques for features extraction, to assess the advantages and disadvantages of pattern modeling techniques specialized for process analysis domain. Finally, we intend to use a larger set of real-world logs as a benchmark for our experiments.

Acknowledgement. This work is partially supported by ITEA3 through the APPSTACLE project (15017) and by the RSA-B project SeCludE.

References

1. van der Aalst, W.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer (2011)
2. van Beest, N.R., Dumas, M., García-Bañuelos, L., La Rosa, M.: Log delta analysis: Interpretable differencing of business process event logs. In: Business Process Management, *LNCS*, vol. 9253, pp. 386–405. Springer (2015)
3. Bose, R., van der Aalst, W.: Abstractions in Process Mining: A Taxonomy of Patterns. In: Business Process Management, *LNCS*, vol. 5701, pp. 159–175. Springer (2009)

4. Bose, R., van der Aalst, W.: Trace clustering based on conserved patterns: Towards achieving better process models. In: Business Process Management, pp. 170–181. Springer (2009)
5. Bose, R., van der Aalst, W.: Discovering signature patterns from event logs. In: Proceedings of Computational Intelligence and Data Mining, pp. 111–118. IEEE (2013)
6. vanden Broucke, S.K., Munoz-Gama, J., Carmona, J., Baesens, B., Vanthienen, J.: Event-based real-time decomposed conformance analysis. In: OTM Confederated International Conferences, pp. 345–363. Springer (2014)
7. Chandrashekar, G., Sahin, F.: A survey on feature selection methods. Computers & Electrical Engineering **40**(1), 16–28 (2014)
8. Chen, N., Hoi, S.C., Xiao, X.: Software process evaluation: A machine learning approach. In: Proc. of Int. Conference on Automated Software Engineering, pp. 333–342. IEEE (2011)
9. Cuzzocrea, A., Folino, F., Guarascio, M., Pontieri, L.: A multi-view learning approach to the discovery of deviant process instances. In: OTM Confederated International Conferences, pp. 146–165. Springer (2015)
10. Diamantini, C., Genga, L., Potena, D., van der Aalst, W.: Building instance graphs for highly variable processes. Expert Systems with Applications **59**, 101–118 (2016)
11. Domingos, P.: MetaCost: A general method for making classifiers cost-sensitive. In: Proceedings of International Conference on Knowledge Discovery and Data Mining, pp. 155–164. ACM (1999)
12. Genga, L., Diamantini, C., Potena, D., Zannone, N.: Deviance mining with concurrency. In: Proceedings of International Workshop on New Frontiers in Mining Complex Patterns (2018)
13. Genga, L., Zannone, N.: Towards a systematic process-aware behavioral analysis for security. In: Proceedings of International Joint Conference on e-Business and Telecommunications, vol. 1, pp. 26–28. SciTePress (2018)
14. Greco, G., Guzzo, A., Manco, G., Saccà, D.: Mining unconnected patterns in workflows. Information Systems **32**(5), 685–712 (2007)
15. Hall, M.A.: Correlation-based feature selection for machine learning. Ph.D. thesis (1999)
16. Han, J., Pei, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufinann (2012)
17. Hwang, S.Y., Wei, C.P., Yang, W.S.: Discovery of temporal patterns from process instances. Computers in Industry **53**(3), 345–364 (2004)
18. Jiang, C., Coenen, F., Zito, M.: A survey of frequent subgraph mining algorithms. The Knowledge Engineering Review **28**(1), 75–105 (2013)
19. Jonyer, I., Cook, D., Holder, L.: Graph-based Hierarchical Conceptual Clustering. The Journal of Machine Learning Research **2**, 19–43 (2002)
20. Kuramochi, M., Karypis, G.: An efficient algorithm for discovering frequent subgraphs. IEEE Transactions on Knowledge and Data Engineering **16**(9), 1038–1051 (2004)
21. Leemans, M., van der Aalst, W.: Discovery of frequent episodes in event logs. In: Proc. of Int. Symposium on Data-driven Process Discovery and Analysis, pp. 1–31. CEUR-ws. org (2014)
22. Lo, D., Cheng, H., Han, J., Khoo, S.C., Sun, C.: Classification of software behaviors for failure detection: a discriminative pattern mining approach. In: Proceedings of International Conference on Knowledge Discovery and Data Mining, pp. 557–566. ACM (2009)
23. Mannhardt, F., Blinde, D.: Analyzing the trajectories of patients with sepsis using process mining. In: Proceedings of RADAR+EMISA, vol. 1859, pp. 72–80. CEUR-ws. org (2017)
24. Molina, L.C., Belanche, L., Nebot, À.: Feature selection algorithms: A survey and experimental evaluation. In: Proceedings of Int. Conf. on Data Mining, pp. 306–313. IEEE (2002)
25. Nguyen, H., Dumas, M., La Rosa, M., Maggi, F.M., Suriadi, S.: Mining business process deviance: a quest for accuracy. In: OTM Confederated International Conferences, pp. 436–445. Springer (2014)
26. Sun, C., Du, J., Chen, N., Khoo, S.C., Yang, Y.: Mining explicit rules for software process evaluation. In: Proceedings of International Conference on Software and System Process, pp. 118–125. ACM (2013)
27. Suriadi, S., Wynn, M.T., Ouyang, C., ter Hofstede, A.H., van Dijk, N.J.: Understanding process behaviours in a large insurance company in Australia: A case study. In: Advanced Information Systems Engineering, *LNCS*, vol. 7908, pp. 449–464. Springer (2013)
28. Tax, N., Genga, L., Zannone, N.: On the use of hierarchical subtrace mining for efficient local process model mining. In: Proceedings of International Symposium on Data-driven Process Discovery and Analysis, *CEUR Workshop Proceedings*, vol. 2016, pp. 8–22. CEUR-WS.org (2017)