Subgraph Mining for Anomalous Pattern Discovery in Event Logs

Laura Genga[1], Domenico Potena[1], Orazio Martino[1], Mahdi Alizadeh[2], Claudia Diamantini[1], and Nicola Zannone[2]

[1] Dipartimento di Ingegneria dell'Informazione, Università Politecnica delle Marche
[2] Eindhoven University of Technology
{c.diamantini,l.genga,d.potena,o.martino}@univpm.it,
{m.alizadeh, n.zannone}@tue.nl

Abstract. Conformance checking allows organizations to verify whether their IT system complies with the prescribed behavior by comparing process executions recorded by the IT system against a process model (representing the normative behavior). However, most of the existing techniques are only able to identify low-level deviations, which provide a scarce support to investigate what actually happened when a process execution deviates from the specification. In this work, we introduce an approach to extract recurrent deviations and generate anomalous patterns from them, which are able to provide meaningful diagnostics of what happened in the system. To detect anomalous behaviors, we apply frequent subgraph mining techniques together with an ad-hoc conformance checking technique. Anomalous patterns are then derived by applying frequent items algorithms to determine highly-correlated deviations, among which ordering relations are inferred. The approach has been validated by means of a set of experiments.

1 Introduction

Organizations are required to monitor their business processes to ensure that their system complies with the prescribed behavior. To this end, organizations usually employ logging mechanisms to record process executions in logs and auditing mechanisms to analyze those logs. Conformance checking has been proposed to assist organizations in verifying whether the observed behavior recorded in an event log matches the prescribed behavior represented as a process model. The notion of *alignments* [1] provides a robust approach to conformance checking, which pinpoints the causes of nonconformity. Given a trace, i.e. a sequence of events generated during a process execution, each representing an instance of a given process activity, and a process model, an alignment maps the trace to a complete run of the model (see [1] for a formal definition of alignment). Take, for example, a trace $\sigma_1 = \langle$customer identification, prepare loan application , check financial status, check external credit rating, check credit purpose, refuse loan$\rangle$ and the loan process in Fig. 1, modeled in the form of a *Petri net*, where boxes represent transitions (denoting process activities) and circles represent places (see [13] for a formal definition of Petri nets). Fig. 2 shows two possible alignments of σ_1 and the net, where activities are abbreviated according to their initial letter(s). The top row of the alignments shows the sequence of events in the trace;

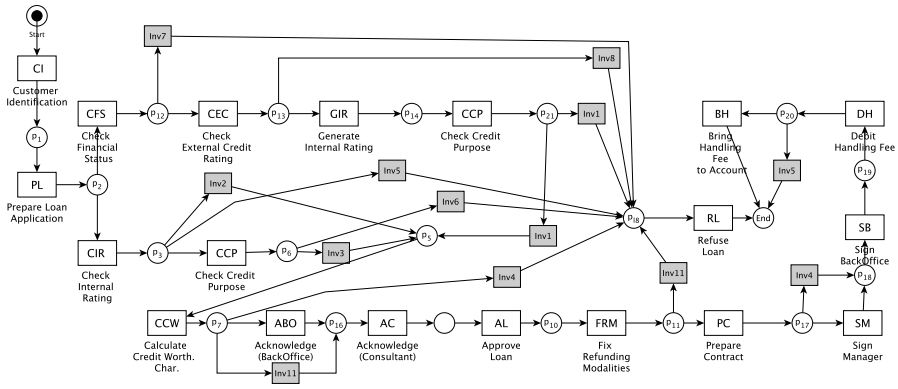


Fig. 1: Loan process represented as a Petri net. The text below the transitions represents the activity label, which is shortened as indicated inside the transitions. Gray boxes represent invisible transitions (i.e. transitions that are not recorded in event logs).

$$\gamma_1 = \begin{array}{|c|c|c|c|c|c|c|} \hline \mathsf{CI} & \mathsf{PL} & \mathsf{CFS} & \mathsf{CEC} & \gg & \mathsf{CCP} & \mathsf{RL} \\ \hline \mathsf{CI} & \mathsf{PL} & \mathsf{CFS} & \mathsf{CEC} & \mathsf{GIR} & \mathsf{CCP} & \mathsf{RL} \\ \hline \end{array} \qquad \gamma_2 = \begin{array}{|c|c|c|c|c|c|c|} \hline \mathsf{CI} & \mathsf{PL} & \gg & \mathsf{CFS} & \mathsf{CEC} & \mathsf{CCP} & \mathsf{RL} \\ \hline \mathsf{CI} & \mathsf{PL} & \mathsf{CIR} & \gg & \gg & \mathsf{CCP} & \mathsf{RL} \\ \hline \end{array}$$

Fig. 2: Alignments of $\sigma_1 = \langle \mathsf{CI}, \mathsf{PL}, \mathsf{CFS}, \mathsf{CEC}, \mathsf{CCP}, \mathsf{RL} \rangle$ and the Petri net in Fig. 1

the bottom row shows the sequence of activities in the run of the net. Deviations are explicitly shown by columns that contain $\gg$. For example, the fifth column in γ_1 shows that an activity must occur in σ_1 according to the net, but it is absent in the trace, i.e. a so-called *move on model*. The fourth and fifth columns in γ_2 show that some events occur in the trace although they are not allowed according to the net, i.e. a so-called *move on log*. Other columns for which events in the trace match the activities in the run of the net represent *synchronous moves*.

As shown in Fig. 2, there can be several (possibly an infinite number of) alignments of a trace and a Petri net, each of them representing a possible explanation of nonconformity. To determine the quality of alignments, a cost is assigned to each move in the alignment. An optimal alignment of a trace and a Petri net according to a given cost function is the one with the least total cost. For instance, if we assign cost 1 to moves on log/model and 0 to synchronous moves, γ_1 is the optimal alignment.

Alignments provide diagnostics in terms of *low level deviations*, i.e. elementary deviations like insertions (i.e., moves on log) and suppressions (i.e., moves on log). While low level deviations indicate where the process deviates, they may not provide meaningful diagnostics. Low level deviations need to be analyzed and correlated together into *high level deviations*, e.g. to show whether an activity has been executed instead of another activities or whether the execution of two activities has been swapped.

However, identifying low level deviations and then using them to diagnose high level deviations has a number of drawbacks. First, it requires analysts reexamining the detected deviations to reconstruct what happened, thus resulting in high operational costs. More importantly, it can lead to inaccurate diagnostics. We illustrate this using the alignments in Fig. 2: γ_1 indicates that activity generate internal rating should

have been executed, whereas γ_2 indicates that activity check internal rating should have been executed and that check financial status and check external credit rating should not have been executed. The low level deviations in γ_2 can be "interpreted" as a high level deviation indicating that check financial status and check external credit rating were executed instead of check internal rating. An analyst can deem this deviation possible and more plausible than a suppression of generate internal rating (i.e., the analyst would choose this replacement as the explanation of nonconformity rather than to the suppression of generate internal rating). As the number of possible alignments can be infinite, existing alignment-based techniques usually return only optimal alignments, i.e. γ_1 in our case. This alignment, however, does not allow the analyst to reconstruct the deemed deviation. The main problem is that optimal alignments are 'optimal' with respect to low level deviations, and it may not be possible to infer what really happens from the moves in these alignments.

To obtain accurate diagnostics, high level deviations should be treated as 'first class citizens' within conformance checking. Adriansyah et al. [3] show how alignment-based techniques can be adapted to explicitly capture high level deviations using anomalous patterns. Intuitively, an anomalous pattern is an artifact representing a behavior that does not comply with the process model. In particular, Adriansyah et al. construct patterns to detect replacements and swaps of (sequences of) activities as Petri nets and show how these patterns can be used to augment a process model. Existing alignment-based techniques can then be applied to the construct alignments that exhibit high-level deviations, providing analysts with accurate diagnostic information.

Although the work in [3] makes a first step toward the detection of high level deviations using alignments, a number of questions are still left open. In particular:

1. *Can we learn patterns representing high level deviations?*
   Adriansyah et al. [3] provide predefined patterns to identify replacements and swaps of activities. We envision that other types of deviations can occur in practice. Analysts may want to identify these deviations in their analysis. This requires defining patterns capturing the desired anomalous behavior. However, the definition of such patterns can be difficult and time consuming. Thus, it is desirable to provide analysts with tool-supported methods for the extraction of anomalous patterns from past process executions.
2. *Which patterns should be considered in the analysis?*
   An analyst might want to recognize any type of deviation in the alignment. However, this significantly increases the search space of alignments, making the approach unpractical. It is worth noting that some anomalous behavior might never occur or be very rare. It is reasonable to ignore these deviations, restricting the attention to recurring anomalous behaviors which are envisaged to occur in the future.

In this work, we address these questions. In particular, the goal is to devise tool-supported methods for the extraction of patterns representing recurrent (complex) anomalous behaviors. To this end, we introduce a novel approach to extract partially ordered anomalous subgraphs. Given an event log and a process model, we apply frequent subgraph mining technique to extract relevant subgraphs and propose a conformance checking algorithm to identify the anomalous ones. Anomalous patterns are derived by detecting correlated anomalous subgraphs by means of frequent itemset algorithms and

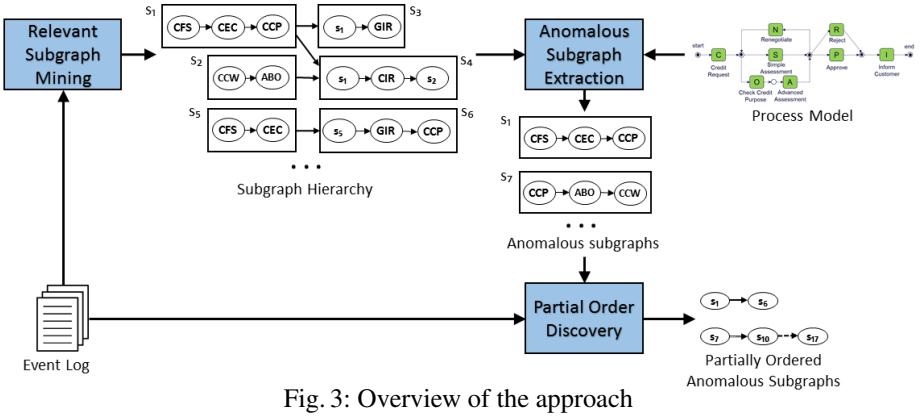


Fig. 3: Overview of the approach

inferring ordering relations among them. The extracted patterns aim to support analysts in conformance checking by providing accurate diagnostics of recurring anomalous behavior, relieving them from the burden of reevaluating situations already analyzed. The approach has been validated by means of a set of experiments.

The remainder of the work is organized as follows. Section 2 details the main steps of the approach; Section 3 presents experimental results; Section 4 discusses related work; finally, Section 5 draws some conclusions and delineates future work.

2 Methodology

Fig. 3 provides an overview of our approach, which comprises three main steps. Given an event log, *relevant subgraphs mining* transforms the log traces into directed graphs and extracts the most relevant subgraphs occurring in the traces using a Frequent Subgraph Mining (FSM) technique. These subgraphs are then analyzed to identify the minimal subgraphs that do not comply with the process model (*anomalous subgraph extraction*). The extracted anomalous subgraphs are used to construct patterns representing frequent (complex) anomalous behavior (*partial order discovery*). In particular, we capture the sets of anomalous subgraphs that frequently occur together by means of frequent itemset discovery algorithms. For each of these itemsets, we infer ordering relations between subgraphs by analyzing their 'position' in the log traces. In the remainder, we describe each step in detail.

2.1 Relevant Subgraph Mining

The first step of our approach aims to mine relevant subgraphs from the process executions recorded in the event log. We transform each log trace σ_i into a directed graph $g_i = (V_i, E_i, \phi_i)$, where V_i is the set of nodes, each corresponding to an event in the trace, E_i is the set of the edges, showing ordering relations among the events, and ϕ_i is a labeling function associating each node to the name of the activity of the corresponding event. For the sake of simplicity, in this work we adopt a simple transformation: a

node is created for each event in the trace and each pair of subsequent events is linked through an edge. By doing so, we consider only the temporal order of events as ordering relations. Note that more advanced strategies can be exploited, e.g. to derive graphs also showing possible parallelisms [12]. We plan to explore these solutions in future work.

To mine relevant subgraphs, we apply a FSM technique, which allows deriving from a given graphs set the set of subgraphs whose *support* (i.e., relevance) is above a certain threshold. In this work, we relate the relevance of a subgraph both to its occurrence frequency and size. Given two subgraphs with the same occurrence frequency but different sizes, we are interested in the largest, since we expect to derive a larger amount of knowledge from it. The size of a graph g can be represented in terms of its Description Length (DL), i.e. the number of bits needed to encode its representation.

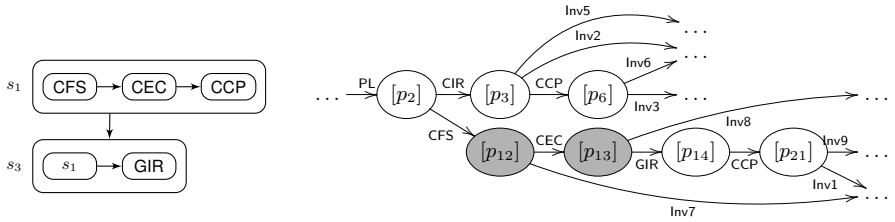
To the best of our knowledge, the only FSM algorithm that explicitly considers DL is SUBDUE [16], which evaluates the relevance of a subgraph in terms of its *compression capability*. Namely, given a graphs set G and a subgraph s, SUBDUE evaluates the DL of G compressed by s, i.e. the dataset obtained by replacing each occurrence of s with a single node. The lower is the DL of the compressed dataset, the higher is the compression capability of s. SUBDUE works iteratively. At each step, it extracts the subgraph with the highest compression capability, which is then used to compress the graphs set. The compressed graphs are presented to SUBDUE again. These steps are repeated until no more compression is possible. The outcome of SUBDUE is a hierarchical structure, where mined subgraphs are ordered according to their relevance, showing the existing inclusion relationships among them. An example of SUBDUE outcome is shown in Fig. 3. Interested readers can find a description of the approach to extract relevant subgraphs from event logs in [11]. The hierarchical structure of subgraphs discovered by SUBDUE becomes the input of the next step of the methodology.

2.2 Anomalous Subgraph Extraction

The second step aims to extract the subgraphs that do not fit the given process model. To this end, we have developed the *Subgraph Conformance Checking* (SCC) algorithm. In contrast to most of the existing conformance checking algorithms, the SCC algorithm is tailored to check conformance of subgraphs corresponding to portions of process executions. The core idea of the SCC algorithm is to replay a subgraph against a process model represented by its coverability graph. Given a Petri net N, the coverability graph of N is a directed graph whose nodes are the markings reachable from the initial marking of N and arcs are labeled by the transitions of N [13].

Given a subgraph s and the coverability graph R of a Petri net, the SCC algorithm identifies all the arcs in R whose labels match with the first activity of s. Starting from each of these edges, the algorithm checks if there exists a sequence of edges in R whose labels match with the sequence of activities in the subgraph. If such a sequence exists, the subgraph is marked as 'compliant'; otherwise, the subgraph is marked as 'anomalous'. The algorithm is robust with respect to the presence of invisible transitions. Edges labeled with invisible transitions are taken into account while exploring the search space, but are not used while matching the paths with the subgraph.

Fig. 4 shows an example of application of the SCC algorithm. Consider subgraph s_1 in Fig. 4a and the (portion of the) coverability graph of the Petri net in Fig. 1 shown



(a) Hierarchical structure of subgraphs to be verified

(b) Portion of the coverability graph of the Petri net in Fig. 1. The states marked in gray are the reachable states.

Fig. 4: Example of application of SCC algorithm.

in Fig. 4b. The SCC algorithm first looks for arcs labeled with the first event in s_1, i.e. CFS, in the coverability graph. There is only one arc labeled with that activity, i.e. $([p_2], [p_{12}])$. The algorithm marks state $[p_{12}]$ as reachable (denoted by gray) and checks whether there exists an arc outgoing that state with label CEC, i.e. the label of the second node of s_1. As the arc exists (i.e., $([p_{12}], [p_{13}])$), also state $[p_{13}]$ is marked as reachable. From $[p_{13}]$, however, there is not any edge labeled with CCP, i.e. the last event of s_1. Therefore, the subgraph is marked as "anomalous".

It is worth noting that among the subgraphs mined by SUBDUE there might occur *inclusion* or *overlapping* relationships, i.e. subgraphs can be completely or partially included in other subgraphs. Let $s_i = (V_i, E_i, \phi_i)$ and $s_j = (V_j, E_j, \phi_j)$ be two subgraphs. We say that s_i *includes* s_j, denoted as $s_i \rightarrow_{incl} s_j$, if (i) $\forall v \in V_j$ there exists $v' \in V_i$ s.t. $\phi_i(v) = \phi_j(v')$ and (ii) $\forall (u, v) \in E_j$ there exists $(u', v') \in E_i$ s.t. $\phi_i(u) = \phi_j(u')$ and $\phi_i(v) = \phi_j(v')$. We say that s_i *overlaps* s_j, if exists a subgraph s_z such that s_i strictly includes s_z and s_j strictly includes s_z.

Given two subgraphs s_i, s_j such that $s_i \rightarrow_{incl} s_j$, it is easy to observe that s_j is able to detect all process executions containing s_i but not the other way around. s_j is hence more general than s_i and thus preferable for the definition of anomalous patterns. To capture this intuition, we introduce the notion of *minimal* anomalous subgraphs. Given a set of subgraphs $S = \{s_1, \ldots, s_n\}$, the set of minimal subgraphs is $S_{min} = \{s_i \mid s_i \in S \wedge \nexists s_j \in S_{min} \text{ s.t. } s_i \rightarrow_{incl} s_j\}$. For the definition of anomalous patterns, we only consider minimal subgraphs. Given the hierarchical structure returned by SUBDUE, we start assessing the conformance of the root subgraphs using the SCC algorithm. If a subgraph is marked as 'anomalous', the algorithm prunes all the branches involving the descendants of the subgraph, since, although they are anomalous 'by inheritance', none of them is minimal. Otherwise, if a subgraph fits the process model, it is marked as 'compliant' and its child subgraphs are iteratively analyzed using the SCC algorithm. The algorithm terminates when all subgraphs in the hierarchical structure are marked as either 'complaint' or 'anomalous'.

Overlapping subgraphs, on the other hand, can provide useful insights about potential anomalous behavior. In fact, two subgraphs overlap on some activities if there are some process executions which differ before/after those activities. By analyzing overlapping subgraphs that frequently occur together, we can explore some portions of these

alternative execution paths. Thus, we consider subgraphs relationships in the final step of the methodology, as explained in the following section.

2.3 Partial Order Discovery

The final step of the approach aims to derive ordering relations among minimal anomalous subgraphs. These ordering relations are used to generate anomalous patterns, i.e. partially ordered subgraphs that show how apparently different anomalous behaviors are usually correlated. First, we generate an occurrence matrix where each cell c_{ij} represents the number of occurrence of the $j\text{-}th$ subgraph in the $i\text{-}th$ trace. We apply well-known *frequent itemset algorithms* [14] to this matrix, thus deriving all the subgraphs which co-occur with a support above a given threshold. To determine how the subgraphs in a frequent itemset are combined, we infer ordering relations between the elements of the itemset pairwise. More precisely, for each pair of subgraphs s_i, s_j belonging to the same itemset, we define one of the following relation: *(i)* the *sequentially* relation, denoted as $s_i \rightarrow_{seq} s_j$, which states that s_j occurs immediately after s_i, *(ii)* the *overlapping* relation, denoted by $s_i \rightarrow_{ov} s_j$, which states that s_i occurs before s_j and their executions overlap, *(iii)* the *eventually* relation, denoted as $s_i \rightarrow_{ev} s_j$, which states that s_j will occur after s_i, but an arbitrary number of other activities (at least one) occured between the two subgraphs. To derive these relations, we analyze the position of the events forming each subgraph of the itemset in the log traces in which the itemset occurs. In particular, we evaluate the occurrence frequency of sequentially, overlapping and eventually relations by means of M_{seq}, M_{ov} and M_{ev} matrices respectively. Each cell of a matrix represents the number of times in which the ordering relation represented by the matrix occurred for a given pair of subgraphs. It is worth noting that in the presence of noisy logs we can detect unreliable relations. To deal with this issue, we consider only ordering relations whose occurrence frequency is above a given threshold.

As an example, let consider the frequent itemset $\{s_{26}, s_{266}, s_{67}\}$ where $s_{26} = \langle \mathsf{CI}, \mathsf{CI} \rangle$, $s_{266} = \langle \mathsf{PL}, \mathsf{CCP} \rangle$ and $s_{67} = \langle \mathsf{CCP}, \mathsf{CIR} \rangle$. Analyzing the positions of subgraphs in the log traces in which these subgraphs occur (see Fig. 5a for an example of such traces), we can observe that s_{266} usually occurs immediately after s_{26} (i.e., $s_{26} \rightarrow_{seq} s_{266}$). Moreover, we can observe that s_{266} overlaps s_{67} (i.e., $s_{266} \rightarrow_{ov} s_{67}$) and s_{67} eventually occurs after s_{26} (i.e., $s_{26} \rightarrow_{ev} s_{67}$). Fig. 5b shows matrices M_{seq}, M_{ov} and M_{ev} for itemset $\{s_{26}, s_{266}, s_{67}\}$. As can be observed in the matrices, these relations are reliable as they have a high occurrence frequency. Fig. 5c shows the obtained partially ordered subgraph.

3 Experiments

We have implemented our approach as two modules of the ESub tool [10], namely *Anomalous Subgraphs Checking* (implementing steps 1 and 2) and *Partial Order Discovery* (implementing step 3).[3] The first module takes as input an event log and the coverability graph of a Petri net and uses SUBDUE to generate a hierarchical structure

[3] `http://193.205.129.67/ESub/GraphManager/browsing_workflow2/subdue`

(a) Excerpt of traces including subgraphs s_{26}, s_{266} and s_{67}

	s_{26}	s_{266}	s_{67}
s_{26}	2	174	0
s_{266}	0	0	0
s_{67}	0	0	0

M_{seq}

	s_{26}	s_{266}	s_{67}
s_{26}	0	0	0
s_{266}	0	0	199
s_{67}	0	0	0

M_{ov}

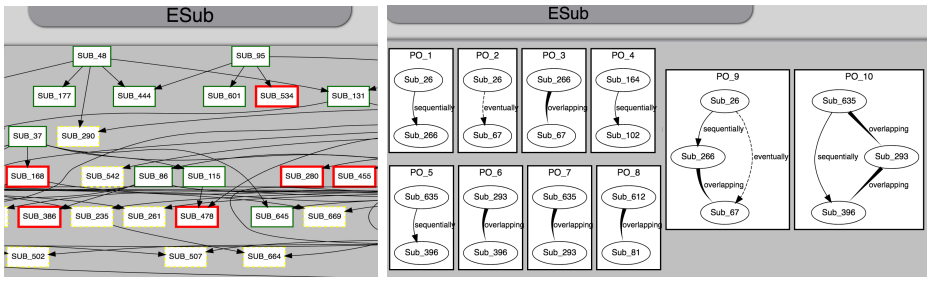
	s_{26}	s_{266}	s_{67}
s_{26}	0	25	199
s_{266}	0	0	0
s_{67}	0	0	0

M_{ev}

(b) Ordering Relations Matrices

(c) Partially Ordered Subgraph

Fig. 5: Ordering Relations Discovery for itemset $\{s_{26}, s_{266}, s_{67}\}$.



(a) Anomalous Subgraphs Checking (b) Partial Order Discovery

Fig. 6: Esub Modules for Anomalous Pattern Extraction.

of subgraphs and the SCC algorithm to extract the anomalous subgraphs. Fig. 6a shows a screenshot of the module displaying a portion of the hierarchical structure derived by SUBDUE where anomalous subgraphs are denoted by a thick border, their children by a dotted border and compliant subgraphs by a normal border. The second module takes as input the set of frequent itemsets and the graphs generated from the log and derives the partially ordered subgraphs. A screenshot of this module is shown in Fig. 6b. Each edge is labeled with the type of relation it represents. Normal lines are used for sequentially relations, bold lines for overlapping relations and dotted lines for eventually relations.

To evaluate the approach we performed a number of experiments using a synthetic event log generated by simulating the Petri net in Fig. 1. This model represents a real-world loan application management process, which has been defined and validated through interviews with the managers of a bank [2]. The results discussed in this section can hence be considered, to a certain extent, representative of the outcome that can be obtained in real-world contexts. Based on this model, we generated 3905 traces consisting of 43673 events using CPN Tools (`http://cpntools.org/`). We manipulated the generated event log by introducing noise. We set for each trace a probability of 20% of having one or more deleted activities and a probability of 20% of having one or more inserted activities, randomly chosen among the activities of the model. In addition, we inserted some high-level deviations, namely *swaps*, *repetitions* and *replacements*. A swap occurs when two or more activities are executed in an opposite order compared to the order defined by the model; we swapped the execution of sequence $\langle$CCP$\rangle$ with the one of $\langle$CIR$\rangle$ in 18.0% of the traces, and the execution of $\langle$GIR$\rangle$ with the execution of $\langle$CFS$\rangle$

Table 1: Support values of discovered partial orders

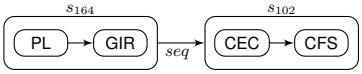
Id	po_1	po_2	po_3	po_4	po_5	po_6	po_7	po_8	po_9	po_{10}
δ_{item}	85.6	96.7	99.8	97.3	96.8	100	99.6	96.4	86.4	99.5
δ_{all}	4.7	5.2	17.9	14.7	11.6	12.2	11.9	8.3	4.4	11.6

in 15.5% of the traces. A repetition means that a given (sequence of) activity(ies) is repeated multiple times (without belonging to a loop); we added two repetitions, namely the repetition of sequence $\langle \mathsf{CI} \rangle$ and of sequence $\langle \mathsf{CER} \rangle$ in 33.6% and 9.0% of the traces respectively. Finally, a replacement indicates that a given (sequence of) activity(ies) is executed instead of another one; in our experiments, sequence of activities $\langle \mathsf{ACO} \rangle$ was replaced with sequence $\langle \mathsf{SBO} \rangle$, $\langle \mathsf{ACO}, \mathsf{AL} \rangle$ with $\langle \mathsf{PLA}, \mathsf{BHF}, \mathsf{GIR} \rangle$ and $\langle \mathsf{PC}, \mathsf{S} \rangle$ with $\langle \mathsf{CCP}, \mathsf{CCP} \rangle$ in 12.0%, 3.7% and 12.0% of the traces respectively.

By doing so, we obtain an event log involving several heterogeneous anomalous behaviors, among which is however possible to recognize some regularities. This reflects what we reasonable expect to find in a real-world context. Note that this implies that we cannot expect to detect patterns with a very high support value. However, this does not affect the validity of the approach. The importance of a deviation is not necessarily related to its frequency. For instance, in several domains (e.g., security), undesired behaviors have to be detected even if they are not very frequent.

The event log was given as input to the Anomalous Subgraphs Checking module. SUBDUE extracted 1245 subgraphs, from which 186 minimal anomalous subgraphs were derived. The set of minimal anomalous subgraphs was used to derive the frequent itemsets. For our experiments, we used FP-Growth [14] implemented in RapidMiner (`https://rapidminer.com`), with a minimum support threshold of 5%. The outcome of FP-Growth algorithm was passed to the Partial Order module. Based on the derived itemsets and a threshold of 40%, the module extracted ten partially ordered subgraphs. Table 1 shows the support of each pattern with respect to the traces where its itemset occurs (δ_{item}) and with respect to the overall set of traces (δ_{all}). We can observe that ordering relations hold for most of the occurrences of their itemsets. Moreover, most of the derived partial orders have good support values, higher than (or anyway closed to) 5%. Note that the support of a pattern is always lower or at most equal to the support of its corresponding itemset. In fact, there can be some traces where the itemset occurs, but its subgraphs do not match the ordering relations of the pattern. This explains why po_1 and po_9 have a δ_{all} lower than 5%. For the sake of space, we only focus on two of the discovered patterns, namely po_4 and po_9, which allow us to point out some interesting aspects of the approach.

Partially ordered subgraph po_4 (Fig. 7a) shows a swap of activities CFS and GIR. The support of the pattern (Table 1) is close with the support we set for the deviation; the difference is due to some traces which do not fit the ordering relations of po_4 because of other inserted/deleted activities. We would like to point out that detecting the swap is quite straight by analyzing po_4 while detecting such a high-level deviation using, for instance, alignments is far from trivial. For example, Fig. 7b shows the alignment returned by the ProM plug-in *PNetReplayer* for trace $\sigma_5 = \langle \mathsf{CI}, \mathsf{CI}, \mathsf{PL}, \mathsf{GIR}, \mathsf{CEC}, \mathsf{CFS}, \mathsf{CCP}, \mathsf{RL} \rangle$. Here, to recognize the swap the analyst has to relate the



(a) Uncompressed po_4

$$\gamma = \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|} \hline \mathsf{CI} & \mathsf{CI} & \mathsf{PL} & \gg & \mathsf{GIR} & \mathsf{CEC} & \gg & \mathsf{CFS} & \mathsf{CCP} & \mathsf{RL} \\ \hline \mathsf{CI} & \gg & \mathsf{PL} & \mathsf{CFS} & \gg & \mathsf{CEC} & \mathsf{GIR} & \gg & \mathsf{CCP} & \mathsf{RL} \\ \hline \end{array}$$

(b) Alignments of trace σ_5 and the Petri net in Fig. 1

Fig. 7: Analysis of anomalous pattern po_4

deletion of CFS and the insertion of GIR, occurring before CEC, with the deletion of GIR and the insertion of CFS occurring after CEC. Clearly, this can easily lead to misleading diagnostics, especially when more than one activity occur between the swapped activities or other deviations occur.

Partially ordered subgraph po_9 (Fig. 5c) corresponds to the combination of two high-level deviations, namely the repetition of activity CI and the swap of activities CCP and CIR, thus originating a new, not a-priori known, pattern. This provides an example of the capability of our approach to extract general patterns that do not necessarily reflect a-priori knowledge of deviations. On the other hand, it also points out the need of post-processing the discovered patterns; it is easy to see that the edge between s_{26} and s_{67} is not needed to interpret the pattern. We plan to address this issue in future work.

4 Related Work

A number of approaches have been proposed for conformance checking. Some approaches [7, 9, 20] check whether log traces satisfy a set of compliance rules. Rozinat and van der Aalst [19] propose a token-based technique to replay process executions over a process model and use the information obtained from remaining and missing tokens to detect deviations. Banescu et al. [6] extend the work in [19] to identify and classify high level deviations by analyzing the configuration of remaining and missing tokens. However, it has been shown that token-based techniques can provide misleading diagnostics.

Recently, alignments have been proposed as a robust approach to conformance checking [1]. Alignments are able to pinpoint deviations causing nonconformity based on a given cost function. These cost functions, however, are usually based on human judgment and, hence, prone to imperfections, which can ultimately lead to incorrect diagnostics. To obtain probable explanations of nonconformity, Alizadeh et al. [4] propose an approach to compute the cost function by analyzing historical logging data, which is extended in [5] to consider multiple process perspectives. Alignment-based techniques rely on total ordering of events; thus, diagnostics obtained by these techniques can be unreliable when timestamps of events are coarse or incorrect. Lu et al. [18] describe how partially ordered traces can be obtained from sequential event logs and propose an approach for computing partially ordered alignments using these partially ordered traces. Alignment-based approaches specify the obtained diagnostic information in terms of low level deviations; our work, instead, is focused on deriving high-level deviations. Adriansyah et al. [3] show how alignment-based techniques can be extended to directly capture high level deviations in alignments using anomalous patterns. However, they use a-priori known patterns, while our approach aims at automatically deriving from the event log recurrent and a-priori unknown patterns.

Well-known approaches for subprocess extraction from sequential traces are [8], which detects subprocesses by identifying sequences of events that fit a-priori defined templates; [15], which exploits a sequence pattern mining algorithm to derive frequent sequences of clinical activities from clinical logs; and [17], which introduces an approach to derive "episodes", i.e, directed graphs where nodes correspond to activities and edges to *eventually-follow* precedence relations, which, given a pair of activities, state which one occurs later. With respect to previous approaches, the one proposed in this work does not require to define any a-priori defined template and extracts the subprocesses that are the most relevant according to the MDL principle, thus taking into account both frequency and size in determining the relevance of each subprocess.

5 Conclusions and future work

In this work, we presented a novel approach to discover complex anomalous patterns, showing high-level deviations in process executions. Main novelties consists in *i)* an approach to extract anomalous subgraphs representing raw deviations and *ii)* an approach to derive partially ordered anomalous subgraphs representing complex anomalous behaviors. Our experiments demonstrated the capability of the approach by returning meaningful patterns capturing high-level deviations that, on the other hand, would be hard to identify using, for instance, alignment-based techniques.

However, more efforts are required in order to move from partially ordered anomalous subgraphs, describing basic ordering relations, to anomalous subprocesses describing the execution flows of deviations. First, a post-processing of the discovered patterns is needed to remove redundant relations, as mentioned in Section 3. Moreover, it is desirable to derive more complex flow constructs, e.g. loops and AND/OR relations. A possible direction in this regard consists in investigating the application of process discovery algorithms. A further extension consists in devising (semi)automatic techniques able to detect in which portions of the process anomalous subprocesses occurred, thus simplifying the analysis of deviations. Moreover, extending the original model with the detected subprocesses paves the way for implementing efficient strategies to detect future instances of anomalous behaviors, both in an on-line and an off-line setting. This can be obtained by investigating the combination of our approach with the one proposed in [3].

In future work, we plan to address these issues. Furthermore, we intend to perform more extensive experiments on real-life event logs, exploring also other approaches, for instance, model building approaches.

Acknowledgement This work has been partially funded by the NWO CyberSecurity programme under the PriCE project and by the Dutch national program COMMIT under the THeCS project.

References

1. van der Aalst, W., Adriansyah, A., van Dongen, B.: Replaying history on process models for conformance checking and performance analysis. Wiley Int. Rev. Data Min. and Knowl. Disc. 2(2), 182–192 (2012)

2. Accorsi, R., Stocker, T.: On the Exploitation of Process Mining for Security Audits: The Conformance Checking Case. In: Proceedings of Annual Symposium on Applied Computing. pp. 1709–1716. ACM (2012)
3. Adriansyah, A., van Dongen, B.F., Zannone, N.: Controlling break-the-glass through alignment. In: Proc. of International Conference on Social Computing. pp. 606–611. IEEE (2013)
4. Alizadeh, M., de Leoni, M., Zannone, N.: History-based construction of alignments for conformance checking: Formalization and implementation. In: Data-Driven Process Discovery and Analysis. LNBIP, vol. 237, pp. 58–78. Springer (2014)
5. Alizadeh, M., de Leoni, M., Zannone, N.: Constructing probable explanations of nonconformity: A data-aware and history-based approach. In: Proceedings of Symposium Series on Computational Intelligence. pp. 1358–1365. IEEE (2015)
6. Banescu, S., Petkovic, M., Zannone, N.: Measuring privacy compliance using fitness metrics. In: Business Process Management. pp. 114–119. LNCS 7481, Springer (2012)
7. Borrego, D., Barba, I.: Conformance checking and diagnosis for declarative business process models in data-aware scenarios. Expert Syst. Appl. 41(11), 5340–5352 (2014)
8. Bose, R.P.J.C., van der Aalst, W.M.P.: Abstractions in process mining: A taxonomy of patterns. In: Business Process Management, LNCS, vol. 5701, pp. 159–175. Springer (2009)
9. Caron, F., Vanthienen, J., Baesens, B.: Comprehensive rule-based compliance checking and risk management with process mining. Decision Support Systems 54(3), 1357–1369 (2013)
10. Diamantini, C., Genga, L., Potena, D.: Esub: Exploration of subgraphs. In: Proceedings of the BPM Demo Session. pp. 70–74. CEUR-WS.org (2015)
11. Diamantini, C., Genga, L., Potena, D.: Behavioral process mining for unstructured processes. Journal of Intelligent Information Systems pp. 1–28 (2016)
12. 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)
13. Finkel, A.: The minimal coverability graph for Petri nets. In: Advances in Petri Nets, pp. 210–243. Springer (1993)
14. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: ACM Sigmod Record. vol. 29, pp. 1–12. ACM (2000)
15. Huang, Z., Lu, X., Duan, H.: On mining clinical pathway patterns from medical behaviors. Artificial intelligence in medicine 56(1), 35–50 (2012)
16. Jonyer, I., Cook, D., Holder, L.: Graph-based Hierarchical Conceptual Clustering. The Journal of Machine Learning Research 2, 19–43 (2002)
17. Leemans, M., van der Aalst, W.: Discovery of frequent episodes in event logs. In: Proc. of Int. Symp. on Data-driven Process Discovery and Analysis. pp. 1–31. CEUR-ws. org (2014)
18. Lu, X., Fahland, D., van der Aalst, W.M.: Conformance checking based on partially ordered event data. In: Business Process Management. pp. 75–88. Springer (2014)
19. Rozinat, A., van der Aalst, W.M.P.: Conformance checking of processes based on monitoring real behavior. Information Systems 33(1), 64–95 (2008)
20. Taghiabadi, E.R., Gromov, V., Fahland, D., van der Aalst, W.P.: Compliance checking of data-aware and resource-aware compliance requirements. In: On the Move to Meaningful Internet Systems. LNCS, vol. 8841, pp. 237–257. Springer (2014)