APD tool: Mining Anomalous Patterns from Event Logs

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

[1] Eindhoven University of Technology
[2] Università Politecnica delle Marche
{l.genga,m.alizadeh,n.zannone}@tue.nl,
{c.diamantini,d.potena}@univpm.it

Abstract. A main challenge of today's organizations is the monitoring of their processes to check whether these processes comply with process models specifying the prescribed behavior. Deviations from the prescribed behavior can represent either legitimate work practices not described by the models, which highlight the need of improving it to better reflect the reality, or malicious behaviors representing, for instance, security breaches and frauds. In this paper, we present a tool designed to extract anomalous patterns representing recurrent deviations, together with their correlations, from historical logging data. The tool is targeted to researchers and practitioners in business process and security domains, with background in process mining.

1 Introduction

Organizations are required to monitor their business processes to ensure that their system complies with the prescribed behavior, typically represented by a *process model*. To this end, organizations usually employ logging mechanisms to record process executions in *event logs*. Event logs consist of *traces*, each of them recording the activities performed in a process execution. Ideally, process executions comply with the defined process models. However, reality may deviate from such models. Deviations can point out the existence of work practices not properly represented by the process model, which hence has to be updated. However, they can also indicate malicious behaviors, like security breaches and frauds, which can lead to severe consequences for an organization, e.g. in terms of loss of money and reputation. It is crucial for organizations to be able to detect and analyze deviations occurred during process executions.

In this paper, we introduce the *Anomalous Pattern Discovery* (APD) tool, whose goal is to assist analysts in exploring *anomalous* behaviors occurred within process executions, i.e. behaviors that do not comply with the prescribed model. More precisely, the APD tool aims to infer *anomalous patterns* showing the **most relevant** anomalous behaviors from historical logging data, together with their **correlations**. Note that in this work we relate the relevance of a deviation to its occurrence frequency. By doing so, APD allows an analyst to focus on recurrent deviations. These deviations are particular interesting as they might indicate work practices (in contrast to isolated incidents) that diverge from the normative behavior prescribed by the organization. Moreover, by exploring correlations among detected deviations the tools allows identifying groups of

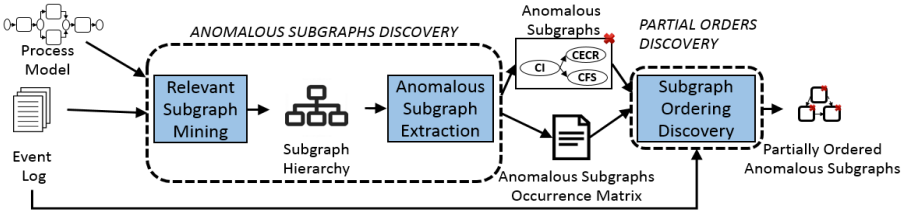

Fig. 1: The Anomalous Pattern Discovery framework

deviations, possibly occurring in different portions of the process, which can be actually considered as part of a single deviant behavior.

It is worth noting that the APD tool can be considered as complementary to other process diagnosis techniques, like, for instance, *conformance checking* [2] that usually focuses on diagnosing single process executions by matching each trace in the log against the process model to pinpoint possible deviations.

The tool has been developed to support the anomalous patterns extraction approach we introduced in a previous work [6]. The approach involves two main steps. Given a process model and an event log consisting of *partially ordered traces* (i.e., traces representing possible parallelisms among process activities, modeled by means of directed graphs), first we extract all anomalous subgraphs, i.e. recurrent subgraphs involving one or more deviations. Then, we generate the set of anomalous patterns representing partially ordered anomalous subgraphs that tend to occur together. The remainder of the paper presents the main functionalities of the APD tool.

2 Tool Description

The APD tool has been implemented as a new plug-in of ESub tool [4], a web application supporting the visualization and exploration of the outcome of subgraph mining algorithms. Fig. 1 provides an overview of the framework underlying the APD framework. The steps of framework have been implemented in two modules, namely the *Anomalous Subgraphs Discovery* module and the *Partial Orders Discovery* module.

The Anomalous Subgraphs Discovery module takes as input *(i)* an event log and *(ii)* a process model, and returns the set of anomalous subgraphs mined from the traces along with an occurrence matrix where each cell c_{ij} represents the number of occurrences of the j-th subgraph in the i-th trace.

The module accepts event logs both in the ".g" format, that is a format used to represent a set of graphs (i.e., in our case, a set of partially ordered traces), and XES format, which is the de-facto standard for event logs. However, since traces in XES format are *totally ordered*, i.e. events are ordered on the basis of their occurrence in the trace, thus hiding possible parallelisms, the module converts XES traces in partially ordered traces by applying the BIG algorithm [5]. Process models are provided in PNML format, which is the standard format to represent Petri nets.

The extraction of anomalous subgraphs involves two steps: *i)* the mining of relevant subgraphs from the event log and *ii)* the compliance checking of the mined sub-

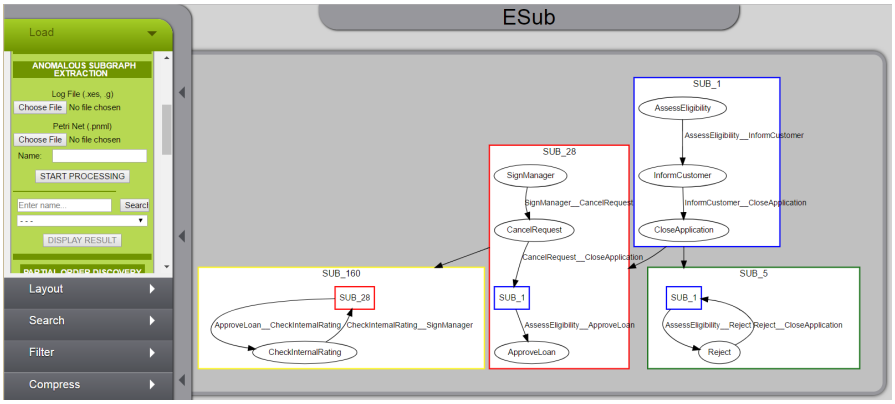

Fig. 2: Anomalous Subgraphs Extraction module

graphs against the given process model. Since traces are represented as directed graphs, we exploit a frequent subgraph mining algorithm to infer the subgraphs (i.e., subprocesses) from the log traces. More precisely, the module exploits the SUBDUE algorithm [8], which extracts and orders subprocesses on the basis of their *Minimum Description Length*, a metric that considers both the dimension and support (i.e., the occurrence frequency) of a subgraph. SUBDUE arranges subgraphs hierarchically, based on their inclusion relations. Specifically, at the top level we have subgraphs that do not include in any other subgraph; whereas descending the hierarchy we have subgraphs built by adding one or more nodes (i.e., process activities and/or subgraphs) to their parent nodes. The module exploits the SUBDUE implementation available at `http://ailab.wsu.edu/subdue/`.

Once the subgraph hierarchy has been generated, the module checks the conformance of each subgraph with the process model. It is worth noting that, when both a subgraph and its children are found anomalous, only the parent subgraph is considered for the analysis. In fact, considering both of them can introduce noise when defining anomalous patterns, since they are strictly correlated because of the inclusion relation. By doing so, we focus on common and more general anomalous subgraphs that are preferable for the definition of anomalous patterns. There are, however, a few exceptions. For instance, we can have anomalous subgraphs with some compliant children. This scenario typically occurs when choice constructs (i.e., XOR) occur in a branch of a parallel behavior as defined in the process model. It is straightforward to observe that a branch without choice constructs will have a higher occurrence frequency than a branch where the choice constructs occur. As a result, a parallel behavior is often captured in the SUBDUE hierarchy by child subgraphs, whose parent subgraph only exhibits a portion of the parallel behavior. Clearly, subgraphs representing a parallel behavior with a missing branch are considered anomalous with respect to the process model. However, these subgraphs may not represent an actual anomalous behavior. In fact, by adding the missing branches they become compliant. Therefore, subgraphs that have (some, but not all) compliant children are neglected in our analysis.

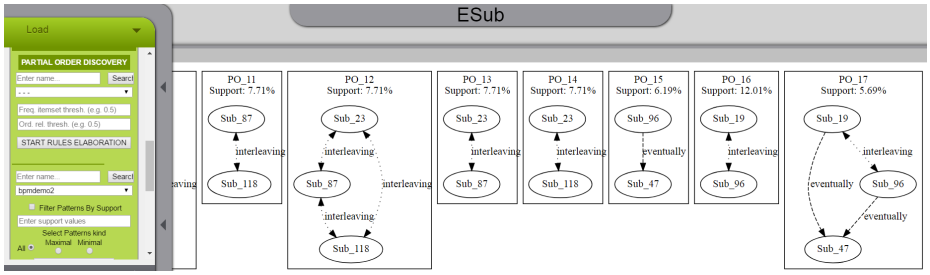

Fig. 3: Partial Order Discovery module

Fig. 2 shows an example of the output returned by the Anomalous Subgraphs Discovery. Compliant subgraphs are denoted by a green dash-dotted line rectangle; anomalous subgraphs are denoted by a red full line rectangle; anomalous subgraphs with some compliant descendants are denoted by a blue dotted line rectangle; finally, subgraphs whose parent is anomalous are denoted by a yellow dash-dotted line rectangle. The set of anomalous subgraphs, together with their occurrence matrix, become the input for the Partial Orders Discovery module.

The Partial Orders Discovery module generates a set of patterns representing partially ordered anomalous subgraphs. First, the module discovers anomalous subgraphs that occur together with a frequency above a user-defined threshold. To this end, a frequent itemset algorithm is exploited. More precisely, the module exploits an implementation of the FP-growth algorithm [7] provided by the SPMF library (`http://www.philippe-fournier-viger.com/spmf/`). Then, the module determines ordering relations among each pair of subgraphs (SUB_i, SUB_j) in the same itemset.

Four types of ordering relations are defined: *i) strictly sequential* relations, which state that SUB_i occurs immediately before SUB_j, *ii) sequential* relations, which state that SUB_i occurs before SUB_j but some activities may (or may not) occur in between, *iii) eventually* relations, which state the SUB_i occurs before SUB_j and at least another activity occurs in between, and *iv) interleaving* relations, which state that some of the activities in SUB_i and SUB_j can be executed concurrently or are shared between the two subgraphs.

To derive these relations, the module analyzes the position of the events forming each subgraph of the itemset in the traces, evaluating the occurrence frequency of each ordering relation for a given pair of subgraphs. Note that we consider only ordering relations whose occurrence frequency is above a user-defined threshold, to deal with possible presence of noise in the event log.

Fig. 3 shows an example of the patterns that can obtained using the APD tool, together with their support with respect to the event log. The tool provides three filtering mechanisms to simplify the exploration of the patterns. In particular, patterns can be filtered on the basis of their support. For example, by selecting a minimum support of 6%, PO_{17} would be removed by the output in Fig. 3. In addition, patterns can be filtered on the basis of their structure. Namely, it is possible to filter all non-maximal patterns (i.e., patterns included in other patterns) or all non-minimal patterns (i.e., patterns including smaller patterns). Finally, the support-based and structure-based filtering can be com-

bined, thus allowing the selection, for instance, of maximal patterns whose support is above a given threshold.

The anomalous patterns extracted using the APD tool can support analysts in various ways. First, they highlight frequent and correlated anomalous behaviors in historical logging data, providing valuable insights to investigate deviant behaviors. In addition, they can be used to enhance classic conformance checking techniques to detect high-level deviations, as proposed in [3]. These patterns can also be exploited for on-line monitoring. In fact, they make it possible to detect the occurrence of recurring anomalous behaviors for which accurate diagnostics is already available when analyzing new process executions, thus relieving the analyst from the burden of reevaluating situations already analyzed. Last but not least, the identified anomalous patterns can drive the definition of measures for preventing and/or responding to deviant behaviors, especially those indicating that the opportunity of a fraud or a security breach exists.

3 Link and Screencast

The ADP tool can be found at `http://kdmg.dii.univpm.it/?q=content/esub`. The tool has been used for the analysis of both synthetic and real-world event logs, including the event log made available for the BPI 2012 challenge [1].

A screencast demonstrating the usage of the APD tool can be found at `https://goo.gl/khpyg9`.

Acknowledgment This work has been partially funded by the NWO CyberSecurity programme under the PriCE project and by the ITEA2 project M2MGrids (No. 13011).

References

1. BPI2012 Challenge. `doi:10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f` (2012), [Online; accessed 07-June-2017]
2. 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)
3. Adriansyah, A., van Dongen, B.F., Zannone, N.: Controlling break-the-glass through alignment. In: Proceedings of International Conference on Social Computing. pp. 606–611. IEEE (2013)
4. Diamantini, C., Genga, L., Potena, D.: Esub: Exploration of subgraphs. Proceedings of the BPM Demo Session pp. 70–74 (2015)
5. 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)
6. Genga, L., Potena, D., Martino, O., Alizadeh, M., Diamantini, C., Zannone, N.: Subgraph Mining for Anomalous Pattern Discovery in Event Logs. In: Proceedings of International Workshop on New Frontiers in Mining Complex Patterns. Springer (2016)
7. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: ACM Sigmod Record. vol. 29, pp. 1–12. ACM (2000)
8. Jonyer, I., Cook, D., Holder, L.: Graph-based Hierarchical Conceptual Clustering. Journal of Machine Learning Research 2, 19–43 (2002)